Titre: Title:	Conception de haut niveau d'une plate-forme SoC et de son système d'interconnexions
Auteur: Author:	Jean Pepga Bissou
Date:	2003
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Pepga Bissou, J. (2003). Conception de haut niveau d'une plate-forme SoC et de son système d'interconnexions [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7148/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7148/
Directeurs de recherche: Advisors:	
Programme: Program:	Unspecified

UNIVERSITÉ DE MONTRÉAL

CONCEPTION DE HAUT NIVEAU D'UNE PLATE-FORME SoC ET DE SON SYSTÈME D'INTERCONNEXIONS

JEAN PEPGA BISSOU

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE EN SCIENCES APPLIQUÉES

(GÉNIE ÉLECTRIQUE)

JUILLET 2003

Your file Votre référence
ISBN: 0-612-86427-8
Our file Notre référence
ISBN: 0-612-86427-8

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE

Ce mémoire intitulé :

CONCEPTION DE HAUT NIVEAU D'UNE PLATE-FORME SoC ET DE SON SYSTÈME D'INTERCONNEXIONS

Présenté par : Jean PEPGA BISSOU

En vue de l'obtention du diplôme de : Maîtrise en sciences appliquées

A été dûment accepté par le jury de l'examen constitué de :

M. Guy BOIS, Ph.D., Président

M. Yvon SAVARIA, Ph.D., membre et directeur de recherche

M. Yves Audet, Ph.D., membre

REMERCIEMENTS

Je souhaite remercier tout d'abord mon directeur de recherche, Yvon Savaria, d'une part pour son soutien financier, mais surtout pour ses conseils toujours avisés. Nos discussions ont toujours été très enrichissantes et m'ont permis de mener ce travail à terme.

J'aimerais également remercier la compagnie Gennum et la Société Canadienne de Microélectronique, pour avoir supporté le déroulement de ce projet. Cette expérience m'a beaucoup apporté du point de vue technique. Je tiens à remercier tous les étudiants du GRM (Groupe de Recherche en Microélectronique) anciens et nouveaux qui ont contribué de manière significative à l'avancement du projet de conception sur les convertisseurs de protocoles réseaux.

Enfin, je remercie ma famille particulièrement mon père et ma feue mère, Bissou Bilong Jonas et Victorine Pepga car sans eux, je ne serais jamais parvenu à cette étape. Sans oublier Christiane Anasatse qui m'a supporté tout au long de ce travail.

RÉSUMÉ

La conception d'une plate-forme SoC pour la conversion de protocoles réseaux a conduit à proposer des architectures spécialisées. Ainsi, il devient possible d'optimiser ces systèmes en séparant les processeurs, les mémoires et les entrées/sorties (E/S). Ces optimisations permettent également d'éliminer les goulots d'étranglement tels que le débit mémoire et l'intégration des E/S.

Une architecture groupant des unités réparties autour d'une mémoire centrale a été conçue dans le cadre de cette recherche. Elle a été validée sur une plate-forme d'évaluation Integrator/AP de la Société ARM. Cette architecture a permis d'analyser le processus de conversion du protocole IEEE 802.3 (Gigabit Ethernet, 1Gbps) au protocole IEEE 1394 (Firewire, b, 800Mbps), et vice versa.

En se basant sur l'étude de ces protocoles et sur les différentes familles d'architectures de processeurs réseaux existantes, nous visons l'élaboration d'une nouvelle architecture plus efficace et mieux adaptée à ce type de tâches. La conception d'une telle architecture nécessite l'utilisation d'une méthodologie de conception de haut niveau. Cette approche mène à la modélisation comportementale et architecturale du système ciblé.

Dans le but d'optimiser cette architecture, nous avons procédé à des simulations sur un modèle comportemental en utilisant un ensemble de vecteurs de tests qui représentent le pire cas d'un flot de paquets à convertir. Ces simulations ont montré que l'architecture

retenue, après l'introduction de modifications significatives, permettait d'atteindre un taux de conversion de 100% du débit admis par les protocoles considérés. Pourtant, l'application de ces mêmes vecteurs sur une architecture de départ validée sur la plate-forme Integrator/AP donnait un taux de conversion de seulement 50%.

Nous montrerons comment la modélisation de haut niveau a permis d'arriver à une nouvelle architecture matérielle/logicielle mieux adaptée. La modélisation architecturale nous a conduit à définir un système d'interconnexion à bus partagés. Cette nouvelle structure basée sur le protocole AMBA est constituée des bus de haute performance de type AHB dont l'un d'eux sert à transférer les flots de données entre la mémoire et les différents modules. Ce dernier a été conçu selon certains requis de fonctionnement dont le plus important est une fréquence d'opération minimale de 320MHz.

ABSTRACT

The design of a SoC platform for network protocols conversion resulted in proposing specialized architectures. It is possible to optimize these systems by separating the processors, the memories, and the inputs/outputs (I/Os). These optimizations also eliminate bottlenecks due to memory throughput and I/O integration.

An architecture that clusters distributed units around a main memory was conceived. It was validated on an Integrator/AP evaluation platform from the ARM corporation. This architecture enables analyzing the process of converting IEEE 802.3 protocol (Gigabit Ethernet, 1Gbps) to IEEE 1394 (Firewire, B, 800Mbps), and vice versa.

Based on the study of these protocols and the various families of existing network processor architectures, we aim at developing a new, more effective and efficient architecture. The design of such architecture requires using a high level design methodology. This approach leads to behavioral and architectural modeling of the targeted system.

In order to optimize this architecture, we carried out simulations on a behavioral model by using a set of tests vectors that represent the worst case of a packet stream to be converted. These simulations showed that the proposed architecture, after the introduction of significant modifications, reached a conversion rate of 100% of the input stream, while the application of these same tests vectors on the initial architecture validated on the Integrator/AP platform gave a conversion rate of 50%. We show how high level modeling enables tuning an efficient software/hardware architecture.

Architectural modeling led us to define an interconnection system with shared buses. This new structure based on the AMBA protocol is composed of high performance buses of the AHB type. One of these buses is used to transfer the data stream between the memory and the various modules. This bus was designed and implemented according to a set of specific requirements. One of the most important is a minimal operating frequency of 320MHz.

TABLE DES MATIÈRES

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT vii

LISTE DES FIGURES xiii

LISTE DES TABLEAUX xvi

LISTE DES NOTATIONS ET SYMBOLES xvii

AVANT-PROPOS xix

CHAPITRE 1 INTRODUCTION 1

1.1 MOTIVATIONS 1

1.2 ORGANISATION DU MÉMOIRE 2

1.3 CONTRIBUTIONS 4

CHAPITRE 2 L'ÉTAT DE L'ART : PROTOCOLES, PROCESSEURS RÉSEAUX ET INTERCONNEXIONS SOC 6

2.1 PROTOCOLES DE COMMUNICATION RÉSEAUX 6

2.1.1 Protocoles de communications 6

2.1.2 Exemples de protocoles réseaux 7

2.1.3 Le problème de conversion de protocoles 11

2.2 FAMILLES DE CONVERTISEURS DE PROTOCOLES 13

2.2.1 Les processeurs réseaux basés sur des processeurs RISC 14

2.2.2 Les processeurs réseaux basés sur des processeurs RISC avancés 17

2.2.3 Les processeurs réseaux basés sur des architectures hybrides 19

2.3 ARCHITECTURE COURANTE DU CONVERTISSEUR 21

2.3.1 Implémentation de l'architecture courante 22

2.4 INTERCONNEXION D'UNE PLATE-FORME SoC 25

2.4.1 Interconnexion à commutation de circuits 25

2.4.2 Interconnexion à bus centralisé 27

2.4.3 Interconnexions à bus partagés 30

2.5 CONCLUSION 31

CHAPITRE 3 CONCEPTION DE HAUT NIVEAU D'UNE PLATE-FORME SOC 33

3.1 PLATE-FORME SoC 33

3.1.1 Le concept de plate-forme SoC 33

3.1.2 Intégration de la plate-forme SoC 34

3.1.3 Prérequis de la plate-forme SoC du convertisseur de protocoles 35

3.2 MODÈLE D'ANALYSE DE LA PLATE-FORME SoC 36

3.2.1 Spécification du système 38

3.2.2 Modélisation comportementale 38

3.2.3 Modélisation architecturale 42

3.2.4 Assignation 47

3.2.5 Raffinement 48

3.2.6 Synthèse logicielle 49

3.2.7 Synthèse matérielle 49

3.2.8 Co-vérification 50

3.3 CONCLUSION 58

CHAPITRE 4 INTERCONNEXION D'UN BUS DE HAUTE PERFORMACE 59

4.1 MODÈLE D'INTERCONNEXION 59

4.1.1 Méthodologie de conception 60

4.1.2 Structure du système d'interconnexion 60

4.1.3 Principe de communication d'un bus AHB 63

4.2 LE BUS AMBA AHB 320 MHz 64

4.2.1 Module maître d'un bus AHB 66

4.2.2 Interface maître du bus AHB 320 MHz 67

4.2.3 Module esclave d'un bus AHB 70

4.2.4 Interface esclave du bus AHB 320 MHz 71

4.2.5 Synchroniseurs d'interfaces 73

4.2.6 Le décodeur du bus AHB 320 MHz 78

4.2.7 L'arbitre du bus AHB 320 MHz 79

4.3 TESTS ET RÉSULTATS 83

4.3.1 Tests applicables 84

4.3.2 Cas de tests 84

4.3.3 Bancs d'essais 86

4.3.4 Résultats 86

CHAPITRE 5 CONCLUSION 92

5.1 SYNTHÈSE DES TRAVAUX 92

5.2 LIMITATIONS ET RECHERCHES FUTURES 94

RÉFÉRENCES 96

ANNEXE 101

ANNEXE 1: CO-VÉRIFICATION DE LA PLATE-FORME SoC DU CONVERTISEUR DE PROTOCOLES RÉSEAUX 102

LISTE DES FIGURES

Figure 2-1 Exemple de communication entre deux entités utilisant le protocole A..... 6

Figure 2-2 Exemple de topologie d'un réseau Firewire 9

Figure 2-3 Exemple de topologie d'un réseau Ethernet. 10

Figure 2-4 : Exemple de communication entre deux réseaux distincts. 12

Figure 2-5 Diagramme bloc du processeur réseau IXP1200 16

Figure 2-6 Diagramme bloc de l'architecture du processeur NP4GS3 18

Figure 2-7 Diagramme bloc du processeur AU1000 20

Figure 2-8 Architecture courante du convertisseur de protocoles réseaux 21

Figure 2-9 (1) Commutations S ; (2) Commutation T 26

Figure 2-10 Diagramme d'un global bus II architecture 29

Figure 2-11 Diagramme bloc d'une micro-PLATcore-7C 31

Figure 3-1 Étapes de conception haut niveau de la plate-forme SoC. 37

Figure 3-2 Diagramme de classes du modèle comportemental du système 39

Figure 3-3 Comparaison du taux de conversion entre l'architecture courante et la nouvelle architecture. 41

Figure 3-4 Structure d'interconnexion pour la nouvelle architecture 45

Figure 3-5 Schéma bloc de la nouvelle architecture du convertisseur de protocoles. 46

Figure 3-6 Communication via le pont AHB/AHB entre bus AHB 40 MHz et AHB 320 MHz. . 53

Figure 3-7 Chronogramme qui illustre le fonctionnement du pont AHB40MHz /AHB 320MHz 53

Figure 3-8 Exemple de communication des esclaves du bus de contrôle AHB 40 MHz 55

Figure 3-9 Taux de couverture du code 57

Figure 4-1 Schéma hiérarchique de la structure d'interconnexion 61

Figure 4-2 Domaines de bus du système 63

Figure 4-3 Chronogramme d'un transfert simple sur un bus AHB 64

Figure 4-4 Schéma global du bus AHB 320 MHz 65

Figure 4-5 Architecture d'un bus AHB à huit maîtres et un esclave 66

Figure 4-6 Architecture d'un maître AHB 67

Figure 4-7 Diagramme bloc d'une interface maître du bus AHB 320 MHz 67

Figure 4-8 Chronogramme de lecture/écriture vue d'un maître du bus AHB 320 MHz 68

Figure 4-9 Architecture d'un esclave AHB 70

Figure 4-10 Chronogramme de l'interface esclave du bus AHB 320 MHz 71

Figure 4-11 Schéma bloc de l'interface des maîtres du bus 320 MHz. 74

Figure 4-12 Schéma logique du synchroniseur d'adresses. 75

Figure 4-13 Schéma logique du synchroniseur de réponse 77

Figure 4-14 Diagramme bloc du synchroniseur de l'interface mémoire 78

Figure 4-15 Architecture d'un arbitre de bus AHB à huit maîtres 79

Figure 4-16 Mécanisme d'arbitrage du bus AHB 320MHz 80

Figure 4-17 Schéma logique de l'arbitre du bus 320MHz. 81

Figure 4-18 Chronogramme de partage de la ressource mémoire. 82

Figure 4-19 Test automatique du bus 320 MHz. 87

Figure 4-20 Illustration du bus AHB 320 MHz qui opère en pipeline. 88

Figure 4-21 Mémoire en mode écriture. 88

Figure 4-22 Mémoire en mode lecture 88

Figure 4-23 Module maître 40 MHz du bus 320 MHz. 89

Figure 4-24 Module maître 80 MHz du bus 320 MHz. 89

Figure 4-25 Synthèse du bus 320MHz 90

Figure 4-26 Dessin des masques du bus 320MHz 91

LISTE DES TABLEAUX

Tableau 3-1 Bandes passantes requises des divers modules. 43

Tableau 3-2 Principales spécifications de la mémoire du système..... 44

Tableau 4-1 Signaux E/S nécessaires d'un module maître du bus AHB 320 MHz 69

Tableau 4-2 Signaux E/S nécessaires de la mémoire 73

Tableau 4-3 Sommaire des différents modules de sélection des synchroniseurs 76

LISTE DES NOTATIONS ET SYMBOLES

AC : Address Converter

AHB : Advanced High Performance Bus

AMBA : Advanced Microcontroller Bus Architecture

APB : Advanced Peripheral Bus

ASB : Advanced Serial Bus

ASIC : Application Specific Integrated Circuit

CISC : Complex Instruction Set Computer

CG : Checksum Generator

CMOS : Complementary Metal-Oxide Semiconductor

CV : Checksum Verifier

DSP : Digital Signal Processor/Processing

FIFO : First In - First Out

FPGA : Field Programmable Gate Array

GF : General Formatter

GRM : Groupe de recherche en microélectronique

IP : Intellectual Property

OSI : Open System Inteconnect

MB : Mailbox

MMAD : Main Memomry Address Distributor

PAx	: Packet Assembler (instance #x)
PRx	: Packet Receiver (instance #x)
RAM	: Random Access Memory
RISC	: Reduced Instruction Set Computer
SoC	: System-On-Chip
SRAM	: Synchronous Random Access Memory
VHDL	: Very high speed integrated circuit Hardware Description Language

- cture où plusieurs flots de calcul parallèles coopèrent sans se nuire.

AVANT-PROPOS

Cette recherche a été effectuée en étroite collaboration avec la Société Gennum, à Burlington, qui s'intéresse au développement d'un système de transmission de données vidéo à travers des réseaux de communications hétérogènes.

L'un des buts d'un sujet de recherche est généralement de trouver, sous une forme ou une autre, une application industrielle. Cette recherche est directement applicable puisqu'elle a été élaborée pour répondre aux besoins formulés par Gennum à savoir comment arriver à transférer des données vidéo entres unités connectés à des réseaux non compatibles.

Ce problème est un travail de recherche, puisqu'il faut premièrement grouper et analyser les protocoles réseaux, déterminer leurs différences fonctionnelles, développer des algorithmes de conversion pour les rendre compatibles et proposer un système d'implémentation de ces algorithmes de manière optimale.

CHAPITRE 1

INTRODUCTION

1.1 MOTIVATIONS

L'utilisation de liaisons fibre optique haut débit permet de nouvelles organisations pour les systèmes multiprocesseurs mémoires partagées. Il devient possible de séparer les processeurs, la mémoire et les entrées/sorties (E/S). Chaque composant peut alors être optimisée, ce qui permet d'éliminer les limitations d'extensibilité tels que le débit mémoire et l'intégration des E/S.

Dans le domaine des communications réseaux, l'utilisation de différents systèmes dans des réseaux de communications distincts, à cause des incompatibilités de leurs protocoles, crée plusieurs problèmes lorsqu'il s'agit de les rendre homogènes. Ces problèmes sont dus principalement à l'évolution des systèmes de communications dans lesquels on utilise une variété de normes et de protocoles de communications souvent incompatibles.

Ainsi, pour permettre aux équipements connectés sur des réseaux distincts et non compatibles de communiquer, il faut ajouter un système entre ces réseaux. Ce système, appelé convertisseur de protocoles réseaux, aura pour but de garantir une communication transparente entre ces réseaux distincts.

Les convertisseurs de protocoles qui existent présentement sont implémentés dans des ASICs ou alors ils sont basés sur des processeurs réseaux [8]. Cependant le désavantage

commun des convertisseurs existants actuellement sur le marché est qu'ils sont spécifiques à une classe restreinte de protocoles de communication réseaux et de ce fait, ils ne sont pas flexibles à des adaptations pour d'autres types de protocoles ou de classes d'applications.

Le but est de concevoir une plate-forme SoC générique pour la conversion des protocoles de communications réseaux. Cette plate-forme devra être optimisée pour le transfert de données vidéo numérique. Cette conception nécessite l'analyse d'architectures spécialisées. Une telle architecturale a été conçue afin de mieux comprendre la conversion du protocole IEEE 802.3 (Gigabit Ethernet)[17] au protocole IEEE 1394 (Firewire b 800MHz) [18]. L'analyse faite après l'implémentation de cette version sur une plate-forme d'évaluation Integrator A/P [4] a permis de redéfinir le problème de la conception de la plate-forme à un haut niveau d'abstraction.

Le problème mentionné revient à développer une méthodologie de conception d'une plate-forme SoC, applicable aux classes d'applications de protocoles de communications réseaux et permettant la production des systèmes dérivés de façon générique. L'environnement de développement d'un tel système constituera la plate-forme SoC générique [4].

1.2 ORGANISATION DU MÉMOIRE

Dans la première partie du chapitre 2, il s'agira de faire une mise en contexte du sujet. Elle débute par une brève présentation des deux familles de protocoles réseaux, Gigabit Ethernet et Firewire, qui seront supportées par le système à concevoir. Par la suite,

l'analyse des architectures supportant des applications de conversion des protocoles réseaux nous amènera à présenter la version courante de notre architecture de convertisseur de protocoles réseaux, qui a été implémentée sur la plate-forme d'évaluation Integrator A/P [4]. Dans la fin de ce même chapitre, nous introduisons le second sujet de ce mémoire, lié à la conception de systèmes d'interconnexions de haute performance utilisables dans l'intégration des modules d'une plate-forme SoC.

Le troisième chapitre sera consacré à la méthodologie de conception haut niveau appliquée au développement de la nouvelle architecture de notre convertisseur de protocoles réseaux. Le but est de développer à haut niveau d'abstraction des modèles comportementaux et architecturaux du nouveau système. Les résultats de ces modélisations permettront d'analyser les performances afin d'optimiser le partitionnement logiciel/matériel du nouveau système.

Suite à cette analyse, une nouvelle proposition d'interconnexions des différents modules a été développée. Ce système d'interconnexion est présenté dans le chapitre quatre. Un élément essentiel de cette approche est l'utilisation d'un bus de communication de haute performance fonctionnant à 320 MHZ. Ce dernier est capable de gérer le flot de traitements des données entre huit modules maîtres et la mémoire de cette architecture.

Ce bus AHB permet aux différents modules d'accéder à la mémoire tout en évitant de créer des goulots d'étranglement sur des flots de traitements. Ce mode d'accès optimal a été implémenté en matériel à partir du protocole AMBA en technologie CMOS 0.18 micron et il fonctionne à la fréquence minimale cible de 320 MHz .

Enfin, le cinquième chapitre servira de conclusion. Nous résumerons l'ensemble des travaux réalisés et présenterons les perspectives futures dans le cadre de cette recherche.

1.3 CONTRIBUTIONS

L'implémentation d'une première version de l'architecture du convertisseur de protocoles réseaux sur la plate-forme d'évaluation Integrator A/P a été le premier objectif de cette recherche. Notre but était d'analyser le fonctionnement des différents modules conçus. Cette évaluation des modules et de l'architecture a démontré la nécessité de faire une analyse à un haut niveau d'abstraction.

Ainsi, les principales contributions de ce travail au niveau recherche en microélectronique sont les suivantes :

- Une étude sur le partitionnement d'une architecture a permis de valider les orientations choisies initialement sur le plan performance et complexité;
- Ceci a contribué au développement d'une seconde plate-forme à partir des modèles comportementaux et architecturaux de haut niveau. Ces analyses ont été comparées aux spécifications recherchées et une approche conceptuelle a été élaborée et validée parmi plusieurs solutions envisageables;
- La recherche d'un système d'interconnexions simple et de haute performance applicable dans l'intégration d'une telle plate-forme SoC, a conduit à la proposition d'une nouvelle structure d'interconnexions à bus partagés basée sur le protocole AMBA qui permet de réaliser des bus de haute performances. L'un des modules de cette structure est un bus AHB supportant une fréquence d'opération de 320 MHz ou plus,

utilisé pour le transfert de données et qui peut servir jusqu'à huit modules maîtres d'une architecture où plusieurs flots de calcul parallèles coopèrent sans se nuire.

CHAPITRE 2

L'ÉTAT DE L'ART : PROTOCOLES, PROCESSEURS RÉSEAUX ET INTERCONNEXIONS SoC

2.1 PROTOCOLES DE COMMUNICATION RÉSEAUX

2.1.1 Protocoles de communications

Par définition, un protocole est un ensemble de règles qui dictent le déroulement d'une activité où les échanges entre les partenaires durant une activité, et qui sont établies sous une certaine convention.

Dans le domaine des télécommunications, on définit un protocole comme étant un ensemble de règles syntaxiques et sémantiques que doivent suivre les entités communicantes pour la transmission des données. Ainsi, les protocoles de communication réseau sont des ensembles de règles utilisables sur des applications d'échange ou de transfert d'information entre les éléments de réseaux de communication.

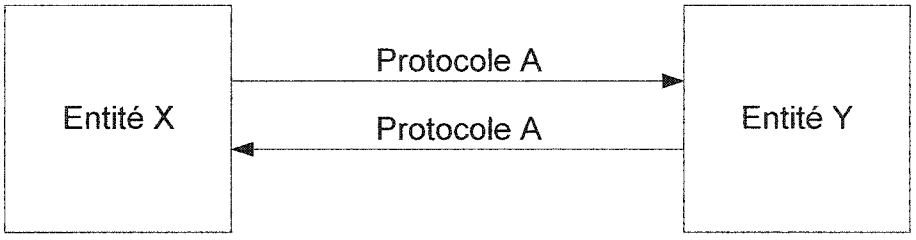

Figure 2-1 Exemple de communication entre deux entités utilisant le protocole A

Le diagramme représenté à la figure 2-1 montre une communication entre deux entités X et Y utilisant le même protocole de communication A. Généralement, ces deux éléments devront appartenir au même réseau de communication.

2.1.2 Exemples de protocoles réseaux

Il existe plusieurs familles de protocoles de communications réseau. Un des objectifs de ce projet est la transmission en temps réel de vidéo numérique de qualité studio sur des réseaux locaux. Il est donc d'intérêt de choisir des protocoles de communication à hauts débits capables de transporter ce genre d'information. Nous souhaitons donc que les réseaux soient en mesure de transporter des charges d'un débit d'au plus 360 Mbps. Dans cette optique, nous devons nous assurer que les protocoles choisis sont en mesure de respecter cette contrainte.

Les formules suivantes permettent de déterminer la largeur de bande brute du réseau afin de permettre la transmission de vidéo numérique d'un signal de 360 Mbps.

- Nombre de paquets/sec = 80% LB du réseau / taille du paquet
- Nombre de paquets/sec = 360 Mbps/charge utile
- Taille du paquet = entêtes + charge utile

Après les substitutions appropriées, on obtient la formule suivante :

- 80% LB = 360 * (entête /charge utile+ 1)

Le débit de transmission tolérable dépend du type d'applications supportés. Dans notre cas nous voulons supporter les applications de vidéo numérique par exemple du HDTV (High Definiton Televison) compressée ou non. Ce dernier utilise une large bande passante (de l'ordre de 1Gbps à 15 Gbps) dont sa limite à 360 Mbps est causée

par l'ajout d'une interface sérielle digitale SDI (Serial Digital Interface) [40] nécessaire à la réception des images par un équipement HDTV. C'est pourquoi dans la formule précédente nous limitons le débit maximal à 360 Mbps.

Dans cette formule, LB désigne la largeur de bande brute du réseau et Entêtes représente la taille des entêtes des paquets. Aussi, nous considérons que les réseaux opèrent à un régime équivalent à 80% de leur bande passante maximale [26]. De cette façon, la probabilité qu'un paquet transmis soit détruit par le réseau est réduite. Cette règle de 80% dépend de la nature du réseau. Ainsi, dans le cas d'un réseau Firewire, le trafic maximum de paquets isochrones (de taille fixe donc le mode de transmission réserve un espace-temps de dimension particulière et cyclique sans correction d'erreur ni de retransmission) ne peut pas dépasser 80% de la bande totale. Le reste des 20% est dédié aux flots asynchrones afin de garantir le transfert des données de contrôles. Cependant dans le cas du réseau Ethernet, il n'y a aucun mécanisme de gestion de bandes passantes par conséquent cette règle ne s'applique pas.

Ainsi, des études ont été faites sur les deux types de protocoles que nous voulons supporter, à savoir Firewire et Gigabit Ethernet. Dans la version courante de l'architecture du convertisseur de protocoles réseaux, il s'agissait de convertir des paquets provenant d'un réseau Firewire vers un réseau Ethernet. La prochaine architecture devra répondre à cette exigence en plus de permettre la conversion dans les deux sens.

Pour investiguer les méthodes de conversion possibles entre ces deux protocoles (Firewire et Gigabit Ethernet), il faut un minimum de compréhension de ces protocoles.

Ce travail ne sera pas couvert en détail dans le cadre de ce mémoire, car il fait l'objet de recherches par d'autres étudiants qui participent au projet qui encadre nos travaux [24]. Néanmoins, nous présentons une vue d'ensemble de ces deux protocoles afin de montrer leurs grandes caractéristiques.

2.1.2.1 ***IEEE 1394 : Firewire***

Le protocole Firewire est une invention de la compagnie Apple. La normalisation de ce protocole s'est faite sous le nom IEEE 1394 [18], qui existe en deux variantes, à savoir le 1394a et le 1394b. Le 1394a est limité à une vitesse de transmission de 400 Mbps avec une trame de taille maximum de 2048 octets. Tandis que le 1394b, plus récent, peut atteindre une vitesse de 3200 Mbps et il supporte une taille maximale de paquets de 32768 octets.

Le réseau Firewire est capable de supporter plusieurs appareils. Comme présenté à la figure 2.2, ces équipements sont diversifiés par leurs applications. Ce type de réseau est conçu généralement pour des environnements audio/vidéo.

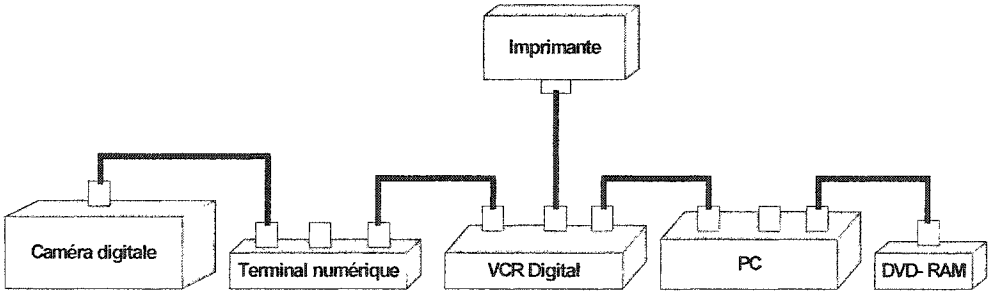

Figure 2-2 Exemple de topologie d'un réseau Firewire

La norme IEEE1394 permet aux entités de communiquer d'une manière dite "peer-to-peer". Ceci nécessite une compatibilité des appareils et surtout qu'une session de communication soit établie avant chaque communication. Cela s'explique par une communication de maître à esclave. La complexité de ce protocole, vu son support des modes de transmission asynchrone et synchrone, nécessite une réduction de son utilisation pour la première application au simple mode de transmission asynchrone.

2 .1.2.2 ***IEEE 802.3 : Gigabit Ethernet***

Ethernet est un protocole très utilisé dans les réseaux LAN. Dans ces réseaux, le transfert des données se fait par le seul bus où sont connectés les entités. C'est pourquoi ce protocole prévoit un mécanisme de contrôle d'envoi simultané de données sur le bus par différents équipements [17]. La figure suivante illustre un exemple de réseau Ethernet.

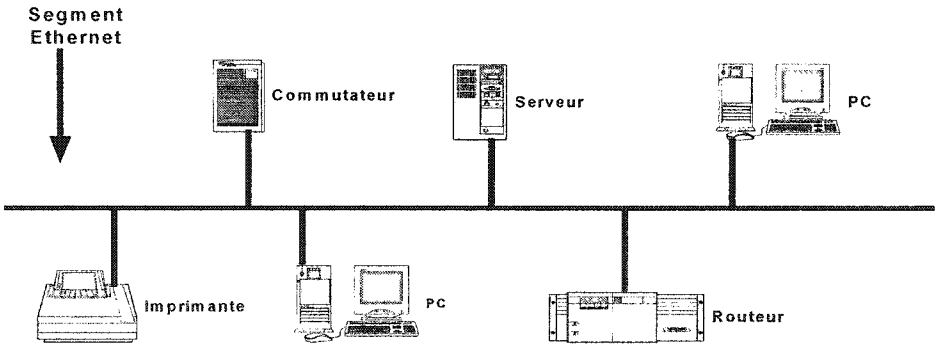

Figure 2-3 Exemple de topologie d'un réseau Ethernet.

Le protocole Ethernet varie selon l'interface physique utilisé. Dans le cas de réseau Ethernet hauts débits, la transmission varie de l'ordre des mégabits jusqu'au gigabits. Un des exemples d'Ethernet haut débit est le Gigabits Ethernet.

2 .1.2.3 ***Différences entre Firewire asynchrone et Gigabits Ethernet***

Les deux protocoles présentés précédemment ont des similitudes, comme par exemple les champs d'adresses source et destination. En revanche leurs différences majeures peuvent être résumées comme suit:

- Firewire utilise un adressage dynamique, alors que les adresses Ethernet sont définies dès la fabrication de la carte réseau de l'équipement;
- L'envoi de paquets asynchrones de Firewire nécessite un accusé de réception; ce mécanisme n'est pas implémenté par Ethernet ;
- Le mode isochrone n'est pas offert par Ethernet;
- Firewire impose d'utiliser un avis de connexion et de déconnexion, ce qui n'est pas demandé par Ethernet.

2.1.3 Le problème de conversion de protocoles

Avant d'expliquer le problème de conversion, rappelons que la conversion des protocoles de communication est un mécanisme qui permet la communication entre systèmes utilisant des protocoles différents. L'implémentation de ce mécanisme nécessite un système de traduction appelé *convertisseur de protocoles*. Il existe plusieurs approches pour résoudre le problème de conversion de protocoles. Cette partie de travail étant déjà traitée dans le cadre d'un projet de recherche est clairement présenté dans [38].

2 .1.3.1 ***Conversion entre deux protocoles distincts***

Supposons que nous avons deux réseaux distincts. Le premier réseau utilise le protocole Firewire non compatible au protocole Ethernet, utilisé dans le second réseau. La figure suivante illustre les deux réseaux.

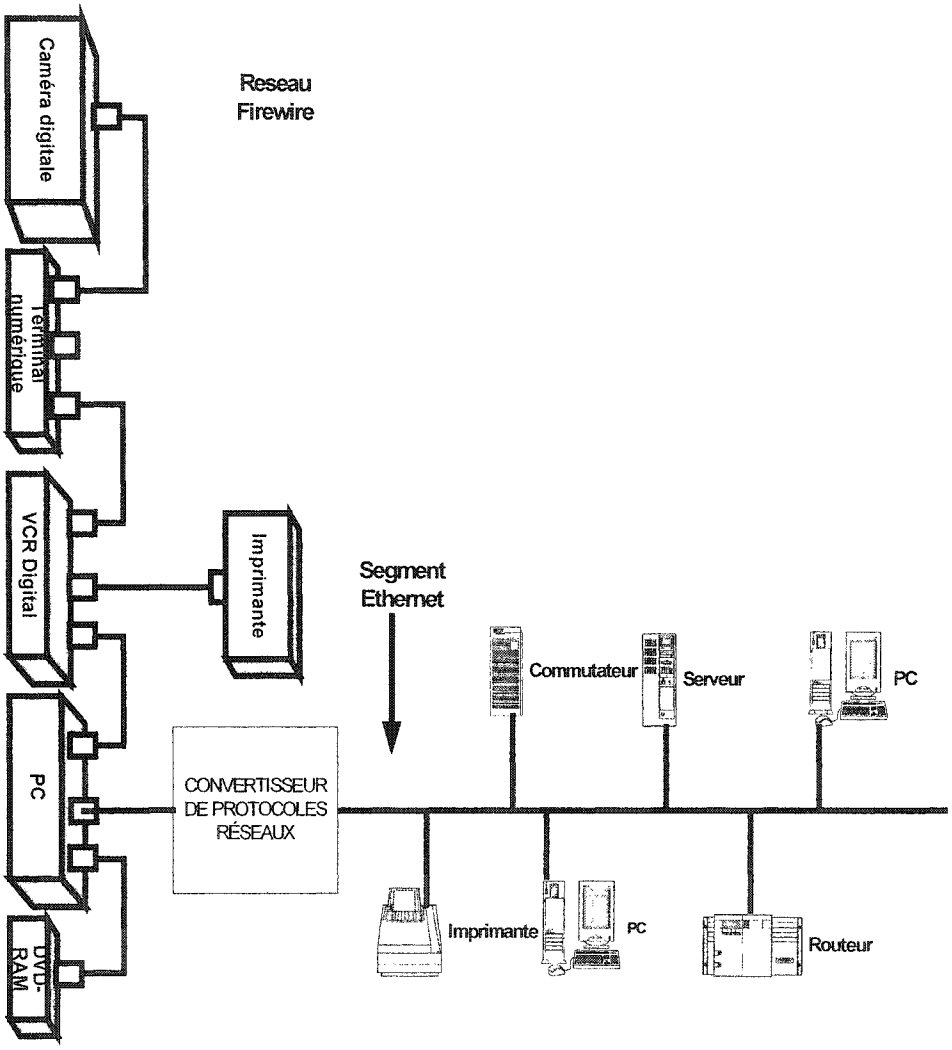

Figure 2-4 : Exemple de communication entre deux réseaux distincts.

Soit un équipement du réseau Firewire qui veut communiquer avec un système PC du réseau Ethernet. Pour cela il faut établir un lien physique entre les deux réseaux à l'aide d'un système appelé *convertisseur de protocoles* qui pourra garantir une telle

communication. Ce système interfacant les deux réseaux devra être capable de comprendre les règles de communications des protocoles Firewire et Ethernet. Cette compréhension lui permettra d'appartenir simultanément aux deux réseaux. Il pourra donc interpréter les messages provenant du réseau Firewire, les convertir en format Ethernet, et les transmettre au réseau Ethernet. Cette opération pourra également être faite dans le sens contraire.

Le convertisseur fait la correspondance à une couche précise du modèle OSI les différents protocoles de communication. Il peut être implémenté au niveau d'un relais ou dans un nœud terminal. Ces différentes techniques d'interconnexions sont expliquées dans [38].

Cette conversion de message d'un protocole à un autre fait intervenir deux types de problèmes:

- Les différences architecturales des systèmes réseaux selon les types d'applications et les protocoles supportés;
- Les différences entres les protocoles de communications en fonction des couches du modèle OSI (*Open System Interconnect*).

Ce dernier problème ne sera pas traité dans ce mémoire car, comme nous l'avons déjà mentionné, il fait l'objet d'une autre recherche [24].

2.2 FAMILLES DE CONVERTISEURS DE PROTOCOLES

Le traitement des paquets, dans le domaine de la communication réseaux, regroupe un ensemble d'applications. Ces applications contiennent diverses fonctions tels que : la

classification ou la modification de paquets, la gestion des files d'attentes, la gestion du service, le routage de paquets, la communication de paquets entre les couches de protocoles et surtout la conversion de paquets d'un protocole à un autre [21].

Actuellement, il existe des systèmes connus sous le nom de processeurs réseaux, dédiés aux applications de traitements de paquets [31]. Ces processeurs réseaux sont conçus selon des architectures complexes et ne sont pas optimisés spécifiquement pour la conversion de protocoles réseaux, mais plutôt pour un ensemble de fonctions applicables aux traitements de paquets. Aussi, aucun des processeurs réseaux existants sur le marché présentement ne supporte le protocole Firewire. Pourtant ce dernier est très répandu, en particulier pour la transmission des données vidéo numériques.

Une étude faite sur certaines architectures de processeurs réseaux existants permet de les regrouper en trois grandes familles architecturales. Présentons les avantages et les inconvénients de chacune d'entre elles.

2.2.1 Les processeurs réseaux basés sur des processeurs RISC

Par définition, un processeur RISC (*Reduced Instruction Set Computer*, ordinateur à jeu d'instructions réduit) est un processeur conçu pour exécuter rapidement une séquence d'instructions simples plutôt qu'un grand nombre d'instructions complexes, comme c'est le cas d'un processeur CISC (*Complex Instruction Set Computer*, ordinateur à jeu d'instructions complexes). Les fonctionnalités habituellement comprises dans les conceptions RISC sont le codage uniforme d'instructions. Par exemple, le code d'opérations possède toujours les mêmes positions binaires dans chaque instruction qui a la longueur d'un mot. Le processeur RISC permet un codage plus rapide avec un

ensemble de registres homogènes utilisables dans n'importe quel contexte. Ceci simplifie la conception du compilateur et des modes d'adressage. Les modes plus complexes sont remplacés par des séquences d'instructions arithmétiques simples [9].

Dans une architecture d'un processeur réseau basé sur des processeurs RISC, l'exécution des opérations est faite par des instructions simples. Ce qui facilite la programmation de ces logiciels et améliore la reconfiguration du système. La majorité des processeurs réseaux de ce type utilisent une configuration architecturale incluant plusieurs processeurs RISC en parallèle. Ceci a pour but d'augmenter la vitesse de traitement et de permettre au système de supporter plusieurs applications opérant à hauts débits. C'est l'exemple du processeur réseau IXP1200 d'Intel [17].

Le IXP1200, l'un des processeurs réseaux de la famille IXP d'Intel, est dédié au traitement de paquets [20]. Ce processeur est conçu sur une architecture matérielle dont les plus grandes caractéristiques sont les suivantes :

- Au cœur du système on retrouve un processeur StrongARM, une amélioration du ARM, qui est un processeur RISC de haute performance utilisé généralement dans les architectures des systèmes embarqués.
- Le traitement rapide des paquets est possible grâce aux six micro-engins [32] travaillant en parallèle. Chaque micro-engin est basé sur une architecture RISC. De plus, c'est un processeur multi-tâches qui supporte quatre tâches (threads) maximum, et qui dispose de huit kilooctets de mémoire d'instructions pour le traitement de paquets.

– Les interfaces physiques des réseaux ciblés sont : Gigabit Ethernet et ATM (Packet Over Sonet et UTOPIA).

Aussi, il dispose d'un ensemble d'interfaces (PC I, UART et Bux IX dual) pour la connexion avec d'autres dispositifs comme l'indique la figure 2-5.

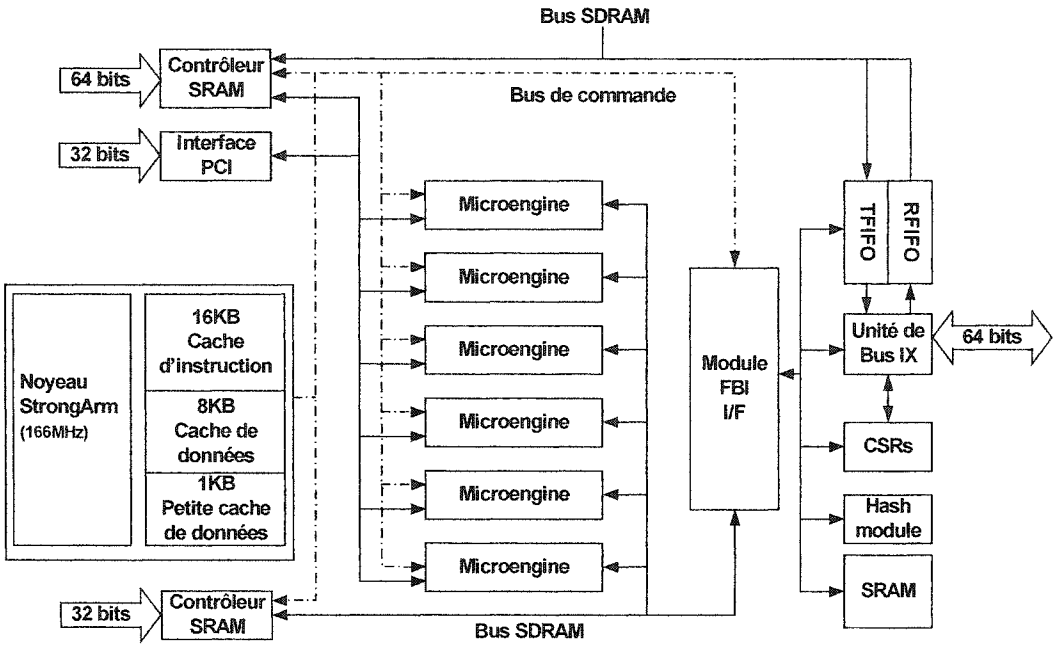

Figure 2-5 Diagramme bloc du processeur réseau IXP1200

Les processeurs réseaux basés sur des RISC sont optimisés généralement pour des fonctions de traitements multicouches à plusieurs protocoles. Ils utilisent plusieurs instructions pour l'exécution d'une simple tâche. En général, le nombre d'instructions augmente en fonction de la complexité de l'application. Ce qui a pour conséquence d'augmenter leurs temps d'exécution. La flexibilité de ce type d'architecture dépend du nombre d'instructions des processeurs.

2.2.2 Les processeurs réseaux basés sur des processeurs RISC avancés

Les processeurs réseaux basés sur des processeurs avancés ont un avantage de solutionner les problèmes des RISC. Ces problèmes sont dus au partitionnement des fonctions dans les modules du système, à l'utilisation de plusieurs processeurs en parallèle (pour d'améliorer les performances du système) et surtout à l'optimisation des interconnections des modules. Le meilleur exemple de ce type de processeur est le processeur NP4GS3 d'IBM [15].

Le NP4GS3 est l'un des processeurs réseaux, de la famille NP d'IBM, utilisé pour les applications de traitements de paquets. Ce processeur est conçu sur une architecture matérielle dont les plus grandes caractéristiques sont les suivantes:

- À la base du système on retrouve un Power PC qui est un processeur RISC avancé conçu par IBM.
- Également, 16 processeurs de protocoles dont chacun est assisté par sept sous-processeurs spécialisés tels que: *Classifier Hardware Assist* (fournit des informations utiles à l'identification des formats de trames), *Tree Search Engine*, *Checksum Coprocessor* ...
- Un coprocesseur pour la recherche d'arbres de routage/filtration/conversion des trames.

Ses interfaces physiques ciblent les réseaux suivants: 10/100 et Gigabit Ethernet et POS (Packet Over SONET). Il dispose d'un ensemble d'interfaces (PC I, UART) pour la connexion avec d'autres dispositifs comme l'indique la figure 2.6.

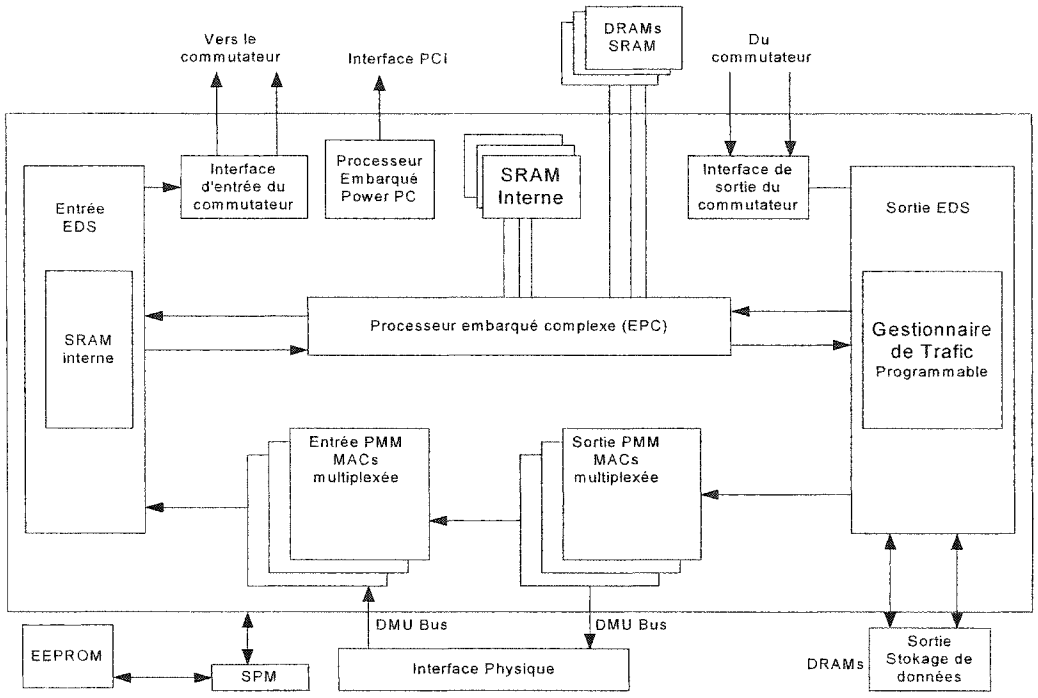

Figure 2-6 Diagramme bloc de l'architecture du processeur NP4GS3

Les processeurs spécialisés utilisent généralement des accélérateurs matériels pour des fonctions complexes. Ceci permet d'augmenter les performances du traitement parallèle. C'est le cas du NP4GS3 qui peut traiter jusqu'à 32 trames en parallèle. Cependant les techniques qui permettent d'augmenter l'exploitation du parallélisme ont un effet néfaste sur la complexité. Plus on augmente son exploitation, plus la complexité de l'architecture augmente. Aussi, les techniques de partitionnement ont une influence sur la flexibilité de l'architecture. Ces processeurs sont généralement optimisés pour le routage des paquets.

2.2.3 Les processeurs réseaux basés sur des architectures hybrides

Les architectures hybrides combinent la programmation des processeurs embarqués avec des coprocesseurs matériels. Ils sont généralement conçus dans l'optique d'être multi-tâches. Ce type d'architecture est flexible pour diverses fonctions et est facilement applicable au trafic de haute vitesse. Un bon exemple est le processeur Alchemy AU1000 d'AMD [1].

Le AU100 est un processeur réseau, construit sur une puce (SoC) dont les principales caractéristiques sont les suivantes :

- Il est doté du processeur MIPS32 et sa fréquence maximale est de 500MHz.
- Il contient un contrôleur DMA à huit canaux et deux contrôleurs d'interruptions (supportant 32 sources d'interruption chacun).
- Il est limité à une consommation minimale, adaptée aux appareils portatifs.
- Les interfaces réseaux disponibles incluent: MAC Ehernet 10/100, USB 1.1 Périphérique et Hôte, UART (Universal Asynchronous Receiver and Transmitter) pour RS232, IrDA (Infrared Data Association), GPIO, AC'97 (Audio CODEC pour ordinateurs personnels), I^2S (Inter IC Sound) et SSI (Synchronous Serial Interface).
- Ces fonctions typiques sont orientées dans les domaines périphériques des réseaux. Ceux-ci pouvant être tant au niveau de l'usager (acces) qu'au niveau des dispositifs des réseaux plus complexe qui requièrent plus de performances (edge). La figure suivante présente la structure de cette architecture.

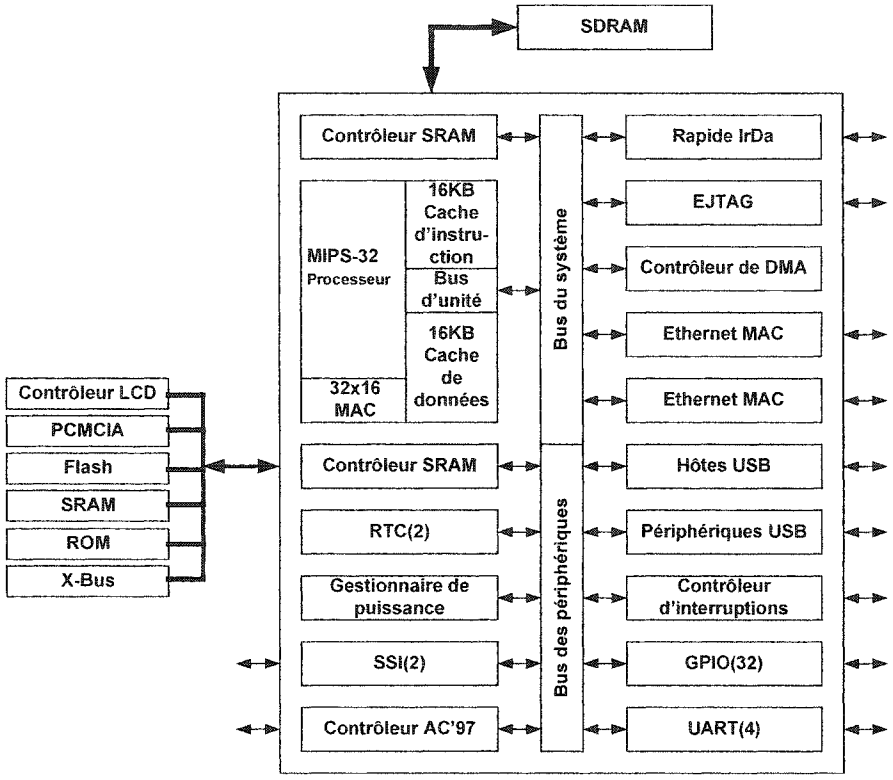

Figure 2-7 Diagramme bloc du processeur AU1000

Les processeurs réseaux basés sur des applications hybrides supportent une bonne plage de vitesse et ne traitent pas la gestion de trafic. L'inconvénient de ce type d'architecture est l'utilisation limitée du processeur central car la grande partie des opérations est gérée par les modules d'accélérations. À moins d'être des IP existants, le temps de développement de ces accélérateurs est long. En plus, il est impossible de modifier une architecture matérielle une fois qu'elle est mise en place dans le silicium.

La version courante de notre architecture de convertisseur de protocoles réseaux se rapproche de la famille de processeurs hybrides.

2.3 ARCHITECTURE COURANTE DU CONVERTISSEUR

La version courante du convertisseur de protocoles réseaux est une architecture basée sur un modèle hybride. Ce système sur une puce (SoC) supporte une conversion de Firewire à Gigabit Ethernet, ce qui n'est pas directement supporté par les processeurs réseaux présentement sur le marché.

Cette architecture, décrite à la figure 2-8, a été développée par un groupe de chercheurs du groupe de recherche en microélectronique (GRM) de l'École Polytechnique de Montréal. Elle regroupe des modules matériels interconnectés et un processeur ARM capable, grâce à un logiciel embarqué, de configurer les coprocesseurs d'identification des protocoles (PI) et de conversions d'adresses (AC) [25].

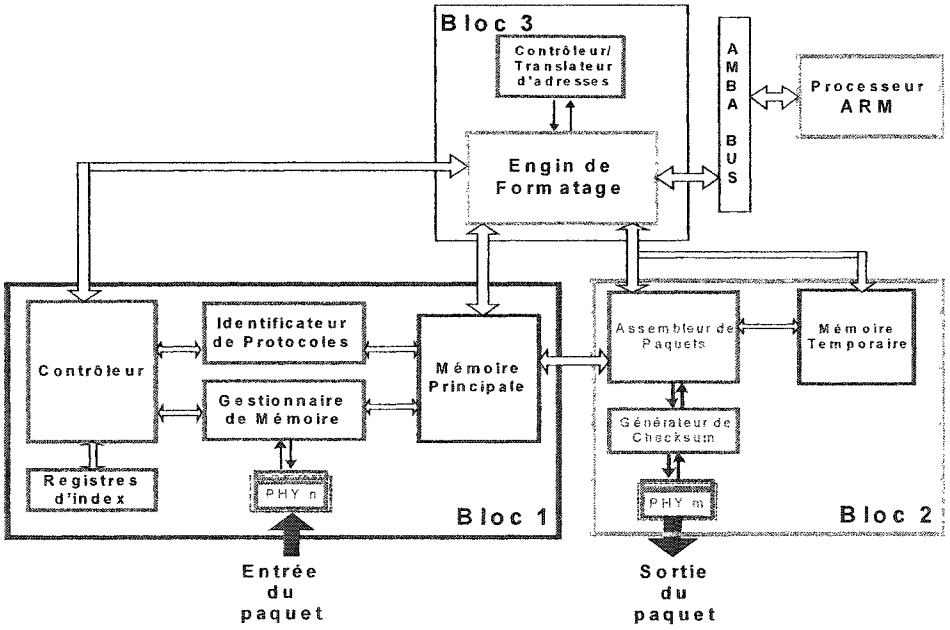

Figure 2-8 Architecture courante du convertisseur de protocoles réseaux

Les spécifications de cette architecture sont limitées à une conversion unidirectionnelle pour faciliter les tests du processus de conversion des protocoles. Ses principales composantes sont:

- Un processeur RISC ARM, des coprocesseurs servant d'accélérateurs matériels chacun étant conçus pour des fonctions précises;
- Deux unités mémoires et un bus interface d'entrée/sortie permettant une connexion à des réseaux Firewire et Ethernet.

Les coprocesseurs ayant fait l'objet de différents travaux de recherche par les étudiants du groupe de recherche ne seront pas présentés dans le cadre de ce travail [11].

L'architecture a été conçue pour être extensible et d'une grande flexibilité en ce qui concerne les protocoles supportables. Cette architecture a été implémentée sur une plate-forme d'évaluation.

2.3.1 Implémentation de l'architecture courante

La première étape de l'implémentation a consisté à se familiariser avec l'environnement ARM-FPGA de la plate-forme Integrator A/P [4]. À l'aide d'un exemple de communication entre une FIFO (First In First Out) implémentée dans le FPGA et le processeur ARM, nous avons, grâce à un simple programme en langage C++ créé une communication entre ces deux modules [27]. On a pu donc lire et stocker des données dans la FIFO via une console. Ensuite, nous avons procédé à l'implémentation de l'architecture courante et à un ensemble de tests nécessaires à la validation du système. Pour ce faire, nous avons commencé par une intégration individuelle des trois blocs de l'architecture et nous avons développé des bancs d'essais sur ces derniers. Nous

avons ensuite assemblé ces trois blocs et procédé aux tests du convertisseur de protocoles en tant que tel. Ainsi, une comparaison avec les résultats obtenus à l'étape de vérification après synthèse, nous a permis de faire une analyse plus détaillée de notre architecture. Cette analyse avait pour but de valider la faisabilité de notre application de conversion avec un tel système.

L'implémentation a été faite dans un FPGA Xilinx, de la famille VirtexE. Chaque bloc de l'architecture a été codé en VHDL (*Very high_speed integrated circuit Hardware Description Language*, Langage de description matérielle pour les circuits intégrés à très grande échelle) [34] en respectant la méthodologie de conception pour la réutilisation [9].

Cette conception hiérarchique de type *bottom-up* (bas vers le haut) consiste à découper le circuit en entités simples pour les intégrer entre elles dans des entités de niveaux supérieurs. Les résultats de cette analyse nous ont permis de détecter certaines déficiences du système.

Ces anomalies se reflètent dans plusieurs domaines à savoir:

- La conversion de flot de paquets Gigabits Ethernet à Firewire: une série de paquets entrants n'est pas totalement convertie à cause de goulots d'étranglement créés par les interconnexions et le faible débit de traitement de certains modules (l'identificateur de protocoles, le formateur et l'assembleur des paquets).
- Le Stockage des données en mémoire : l'utilisation de plusieurs mémoires pour le stockage de données des paquets est nécessaire si le système de gestion du flux est optimisé [14].

– L'interconnexion des modules. l'implémentation d'un mode de communication synchrone entre modules opérant à des fréquences différentes et ayant des délais de traitement variés, nécessite l'insertion des tampons (FIFO) aux entrées et sorties des modules. Ces tampons devront avoir une profondeur définie par le pire cas d'une conversion, ce qui rend nécessaire une analyse à haut niveau.

– La conversion unidirectionnelle: la conversion dans cette architecture n'est possible que dans un sens; à savoir de Gigabit Ethernet à Firewire. Une conversion bidirectionnelle peut être envisagée avec la possibilité de supporter des traitements parallèles provenant simultanément des ports de communication Ethernet et Firewire.

– La gestion du système : le contrôleur du système est une implémentation matérielle. Mais cette implémentation peut aussi être faite en logiciel sans perte de performance au niveau système. Ainsi le processeur ARM, dont le taux d'utilisation du CPU est très faible sous certaines conditions, pourra être utilisé entre autre comme outil de configuration et de contrôle du système.

– Le faible degré de flexibilité: si une légère modification d'un module de cette architecture a des répercussions au niveau du fonctionnement global du système, il faut envisager une nouvelle structure d'interconnexion plus flexible et un contrôleur (re)programmable [33].

L'implémentation de cette architecture courante sur la plate-forme d'évaluation Integrator A/P a permis de démontrer que le système ne pouvait garantir un traitement de flots de paquets en temps réel avec une latence acceptable. Ceci est principalement dû à certaines lacunes tels que le traitement séquentiel des paquets, sa faiblesse au niveau

de l'optimisation du transfert des données entre modules internes et aussi son manque de flexibilité.

L'amélioration des requis du système a conduit à l'élaboration d'une nouvelle architecture plus complexe et nécessitant une étude à un haut niveau d'abstraction. L'un des éléments nécessaires à l'élaboration de cette nouvelle plate-forme SoC est son module d'interconnexion, qui doit faciliter la communication interne entre les modules du système.

2.4 INTERCONNEXION D'UNE PLATE-FORME SoC

Cette partie présente trois modèles d'interconnexions pour des plates-formes SoC. Le premier modèle a été conçu pour l'optimisation des traitements numériques de signaux vidéo. Le second modèle, moins complexe que le premier, est une approche plus avancée. Le troisième est l'une des méthodes les plus communes. Nous allons étudier les principes de chacun et ressortir leurs avantages et inconvénients.

2.4.1 Interconnexion à commutation de circuits

Les principes de la commutation de circuits par une approche dite TST consiste à cascader un étage de commutation temporelle (T) effectuée par des tampons, un étage de commutation spatiale (S) dans un *Crossbar* (commutateur de canaux physiques) et enfin un étage de commutation temporelle (T) [12].

Le choix de conception d'une architecture TST est décrit dans [23]. La figure suivante donne des exemples de commutations S et T.

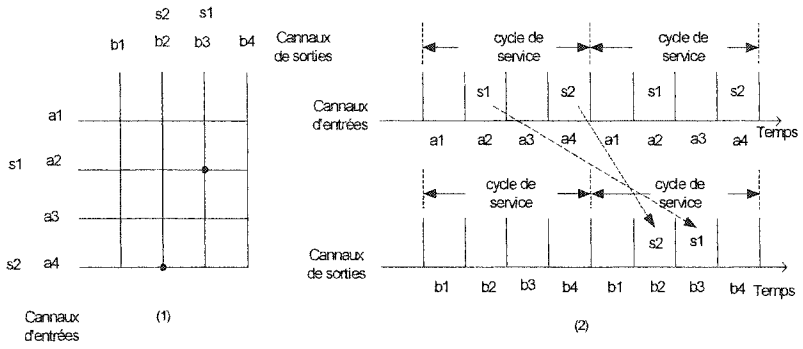

Figure 2-9 (1) Commutations S ; (2) Commutation T

La commutation en espace S est l'allocation de points de contact dans le crossbar pour des intervalles de temps déterminés. C'est un problème très complexe et une approche consiste à décaler dans le temps, au moyen de tampons d'entrées sorties, les flux de manière à éliminer les goulots d'étranglement. Plusieurs méthodes existent à cet effet et une d'elles est décrite par le théorème de Konig-Hall [13].

La commutation T divise le temps en périodes de servie. Ces dernières sont divisées en tranches de temps. Ainsi, on peut réaliser un canal T par l'allocation d'une ou de plusieurs tranches de temps dans la période de service.

Un élément important de cette technique est lié à la nécessité de prévoir, dans le pire cas de l'inversion de l'ordre des blocs de données, une capacité de mémorisation de 50% de la plus longue période de service.

L'un des exemples récents de cette application est le PROPHID qui a été conçu en 1996 par la compagnie PHILIPS pour des interconnexions à haut débits [12]. En effet, il permet la connexion de plusieurs modules (processeurs et coprocesseurs) orientés dans le transfert des données, afin de former un système hétérogène. Les communications

sont unidirectionnelles et configurables. Elles utilisent un contrôleur de flux de type FIFO.

Nous constatons qu'un système TST doit être accompagné d'un autre système qui permet l'adressage des processeurs, pour la configuration et surtout le contrôle. En plus, il est possible qu'il y ait des tampons volumineux et dans ce cas, il faut prévoir une mémoire externe. Ce qui n'est pas souhaitable pour une plate-forme SoC.

Les principaux avantages du TST pour une plate-forme SoC sont sa grande bande passante, son extensibilité et son fonctionnement déterministe qui garantit un taux de disponibilité de 100% car chaque canal offre une bande passante réservée d'avance et garantie durant le transfert.

Le PROPHIB manque de flexibilité, les coûts des *crossbars* et surtout leurs tables de contrôle croissent à la puissance n^2 pour n processeurs utilisés.

2.4.2 **Interconnexion à bus centralisé**

Généralement, les bus sont classés en bus de contrôle, entre unité de contrôle (UC) et mémoire, et en bus d'entrées et sorties (E/S). Les bus d'E/S peuvent être longs et connecter plusieurs types de composants avec des gammes étendues de débits en satisfaisant à une norme. Cependant, les bus UC-mémoire sont généralement courts et très rapides. Ils sont utilisés principalement pour l'optimisation de la bande passante entre l'UC et la mémoire. Mais leur conception nécessite une connaissance de tous les types de composants qui doivent s'y connecter.

Une transaction typique d'un bus interne comprend deux parties : une phase d'adresse et une phase de donnée. Ces transactions sont généralement définies du point de vue

mémoire pour une lecture comme étant un transfert de données depuis la mémoire vers un composant ou d'une écriture dans le cas contraire.

Dans chacune de ces transactions, l'adresse est d'abord envoyée à travers le bus vers la mémoire ainsi que les signaux de contrôles indiquent le type d'opération à effectuer. La mémoire répond en renvoyant (dans le cas d'une lecture) ou en recevant (dans le cas d'une écriture) la donnée impliquée dans la transaction. Dépendamment du protocole du bus, l'adresse et la donnée peuvent être transmises et validées au même moment en écriture.

Dans le cas d'un système à plusieurs processeurs, l'efficacité de l'utilisation d'un bus centralisé dépend de l'architecture du bus de communication utilisé. Plusieurs études ont été faites à ce sujet dont l'une d'elles a permis de faire une comparaison sur les différents bus utilisés dans la conception de plate-formes SoC [29]. Cette étude compare l'exécution d'une application d'émetteur et de décodeur MPEG2 utilisant une transmission OFDM (Orthogonal Frequency Division Multipexing). Elle conclut que chaque architecture peut de manière significative, augmenter l'exécution si les opérations sont canalisées sur certains nœuds de calculs. Cependant, l'exécution d'un système global peut être limitée par des goulots d'étranglement selon le type d'architecture de bus utilisés par rapport aux modules de calculs. Parmi ces cinq architectures de bus, les tests de comparaison ont prouvé que le GBIIA (Global Bus II Architecture) surpasse de loin les performance des autres types.

Le GBIIA est une architecture utilisant un bus centralisé comme l'indique la figure 2-10. Cette architecture permet le partage d'un bus général avec un protocole d'arbitration et

des techniques de signalisation des états des données semblables à une architecture de Crossbar. Les sous bus du système sont considérés comme étant des nœuds de calculs (Compute Node) CN chacun doté d'un processeur et d'une mémoire SRAM. Dans [29] une des SRAM (D) contient vingt-quatre espaces mémoires pour mémoriser les fanions.

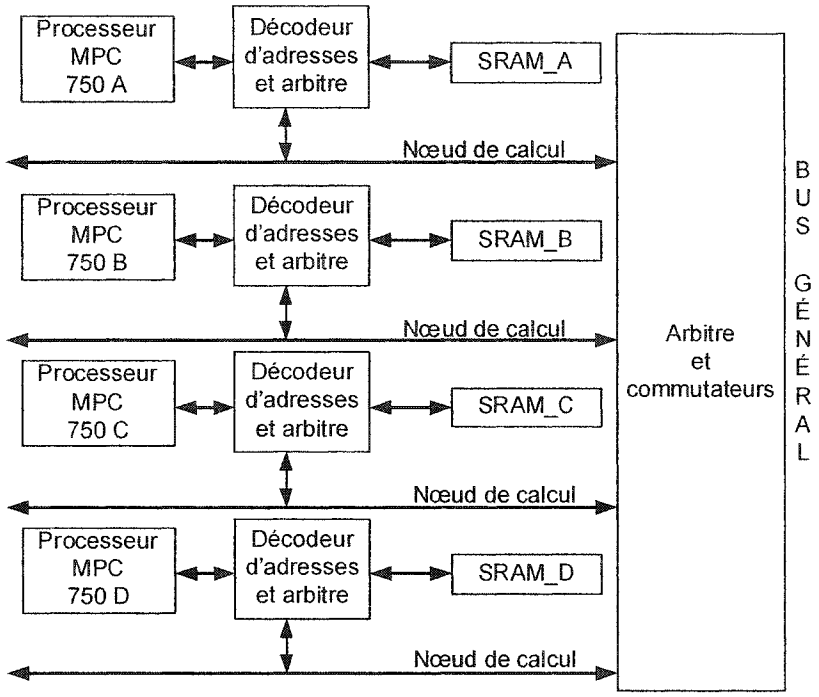

Figure 2-10 Diagramme d'un global bus II architecture

Le bus global permet une seule communication à la fois vers les CN. Ainsi, tous les modules ne peuvent pas être accédés sans latence, comme le permet sous certaines conditions le bus de communication d'IBM CoreConnect (CCBA) [15]. Ce dernier possède trois niveaux hiérarchiques, à savoir : le Processor Local Bus (PLB), le On-chip Peripheral Bus (OPB) et le Device Control Register (DCR). Cependant, son implémentation est laborieuse, contrairement au bus AMBA [3] qui est plus flexible.

2.4.3 Interconnexions à bus partagés

Le concept d'interconnexion à bus partagés consiste à reprendre les bus conçus pour des cartes imprimées et châssis (PCI, ISA...) et les transposer à des systèmes sur une puce (SoC). Plusieurs nouvelles architectures de bus ont été développées pour surpasser les limitations des bus. L'exemple du peripheral Interconnect Bus (PI-Bus) est une architecture de bus pour systèmes monolithiques, normalisée par plusieurs industriels vers les années 1990. Il est utilisé dans un très grand nombre de produits et de ce fait, presque tous les systèmes intégrés ont recours à des technologies équivalentes ou similaires, soit sous la forme de technologies propriétaires ou sous la forme de normes comme AMBA [3].

Le bus AMBA est utilisé dans plusieurs champs d'application et il permet l'intégration de différentes plates-formes sur une puce SoC, d'où le nom *AMBA-based uP Sub-system* [36]. Aussi, il est possible d'avoir plusieurs IP AMBA dans la conception d'un système. Un exemple de cette implémentation est la micro plate-forme uPLAT [22]. La figure 2-11 présente l'architecture de ce système. Elle permet la réutilisation des modules ainsi que l'obtention d'éléments essentiels pour une intégration rapide d'une plate-forme SoC de grande taille.

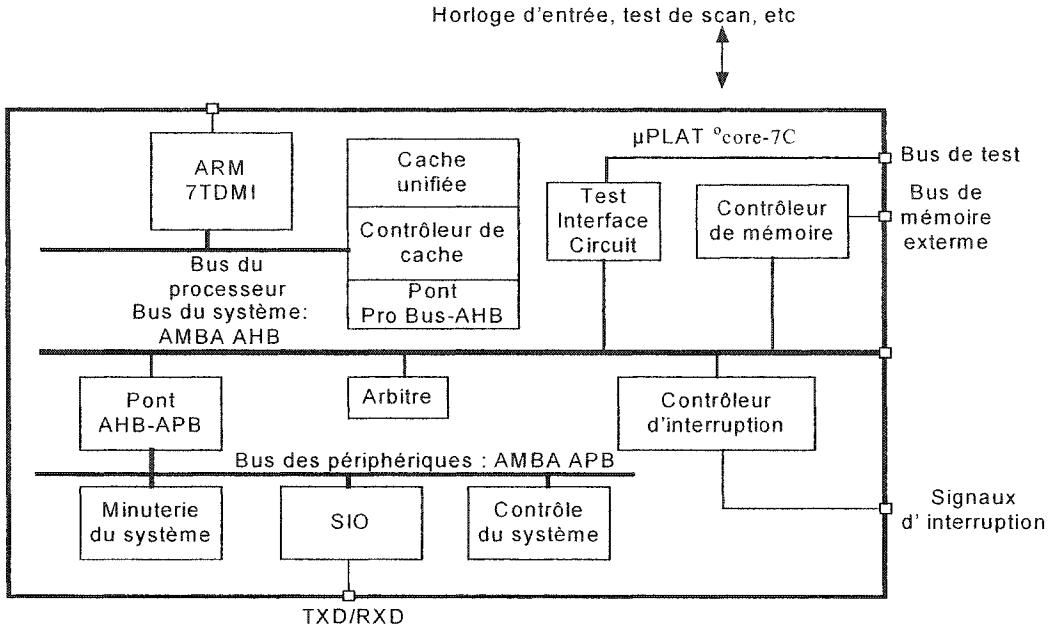

Figure 2-11 Diagramme bloc d'une micro-PLATcore-7C

Comme la conception d'une plate-forme SoC nécessite un temps de développement très important, surtout pour des systèmes complexes, la réutilisation des modules permet de réduire ce temps de conception. Aussi, si le module élémentaire dans un système est l'exécution d'une fonction de base comme dans le cas d'un système d'exploitation en temps réel, l'intégration d'un système d'interconnexion simple et rapide à implémenter permet d'augmenter la flexibilité d'une telle structure au niveau de ces applications [33].

2.5 CONCLUSION

Dans une première phase, une architecture de base du convertisseur de protocoles réseaux regroupant des unités réparties autour d'une mémoire centrale a été conçue. Cette architecture a été testée sur la plate-forme d'évaluation Integrator A/P. L'analyse fonctionnelle de cette implémentation a été faite afin de bien étudier les besoins réels d'un système de conversion de protocoles. Les résultats de cette analyse ont permis de

mieux comprendre les étapes nécessaires pour la conversion d'un protocole Firewire à un protocole Gigabit Ethernet.

En partant de l'étude des protocoles Firewire et Gigabit Ethernet, et en considérant différentes familles d'architectures de processeurs réseaux, une nouvelle architecture devra être construite afin de remédier aux limites de la version courante de notre système. L'aboutissement à une architecture efficace nécessite l'utilisation d'une méthodologie de conception de haut niveau.

CHAPITRE 3

CONCEPTION DE HAUT NIVEAU D'UNE PLATE-FORME SoC

Le but est de définir un cadre de travail constitué d'un ensemble d'outils de conception, de bibliothèques matérielles/logicielles et d'un flot de traitements facilitant la conception générique d'une architecture de convertisseurs de protocoles réseaux, qui répondent aux requis prédéfinis. Cet environnement constitue la *plate-forme SoC générique pour la conversion de protocoles réseaux*.

3.1 PLATE-FORME SoC

3.1.1 Le concept de plate-forme SoC

Le concept de plate-forme SoC est d'actualité depuis quelques années. Cependant, plusieurs définitions du concept de plate-forme ont été utilisées dans le contexte de la conception des produits électroniques. Ces définitions semblent aussi dépendre du domaine d'application [1]. Ainsi, il existe plusieurs classes de plate-forme à savoir : les plates-formes matérielles, les plates-formes logicielles, les plates-formes d'ordinateurs, des produits dérivées et des interfaces standards de plates-formes [33].

Dans le domaine des circuits intégrés, une plate-forme se définit comme étant un circuit intégré flexible, dédié à un ensemble d'applications prédéfinies, dont la configuration ou la reconfiguration est réalisable par une simple programmation ou une configuration d'une ou de plusieurs composantes de la puce (SoC) [8]. L'aspect

programmation implique une modification électrique ou l'utilisation de logiciels exécutés sur un microprocesseur [35]. Le problème avec cette approche est le manque potentiel d'optimisation qui peut provoquer une baisse de performance ou une augmentation de la surface physique de la puce [30].

Nous utilisons le terme plate-forme pour décrire un environnement architectural créé afin de faciliter la réutilisation des modules nécessaires à la conception ou aux développements des applications d'un système sur puce optimisé pour la conversion de protocoles.

Dans un marché influencé par les consommateurs, les projections raisonnables démontrent que dans quelques années, nous devrons tenir compte du facteur réutilisation des modules pour arriver à satisfaire les objectifs de productivité [9]. Pour atteindre la productivité visée, sans toutefois nuire à l'utilisation efficace du silicium, la réutilisation devra être facilitée. Ceci est facilité si on considère des applications spécifiques. D'autre part, la productivité est aussi facilitée par l'utilisation de connexions virtuelles (*virtual sockets*) dans lesquelles peuvent être introduits des composants virtuels [8] [5].

3.1.2 Intégration de la plate-forme SoC

Comme l'industrie de la conception des systèmes intégrés demande une rapidité de mise en marché, des produits plus flexibles, avec des systèmes de plus en plus complexes et comprenant certains risques de conceptions et exigeant des coûts faibles, l'élaboration d'une plate-forme générique procure plusieurs avantages [41].

L'intégration d'une plate-forme comporte de nombreuses facettes comme les méthodologies de design, la modélisation des standards, la caractérisation d'une bibliothèque de composantes virtuels, la conception/classification d'une bibliothèque de logiciels dédiés, la structure matérielle, la vérification logicielle et matérielle et le prototypage des systèmes sur puce [8] [5].

Il est important de souligner que l'idée d'intégration est surtout centrée sur la structure matérielle du système, car celle-ci permettra aux applications qui devront être implémentées d'accomplir leurs multiples fonctions dans un environnement précis. Le cœur de la conception matérielle est basé sur les différentes applications que supportera le système. On pourra donc contrôler l'ensemble des composants virtuels du système une fois que notre environnement matériel sera bien défini.

3.1.3 Prérequis de la plate-forme SoC du convertisseur de protocoles

La réalisation d'un système matériel qui supporte des applications logicielles, basées sur des prérequis divers, comme pour notre convertisseur de protocoles, apporte des contraintes supplémentaires [28]. Ainsi, pour l'intégration de notre convertisseur, il serait important de faire une étude approfondie sur les différents aspects suivants :

- l'analyse des protocoles de communication réseau supportés et envisageables;
- l'étude des algorithmes de conversion des protocoles de communications réseaux;
- l'analyse des étapes nécessaires à l'acheminement des données dans un convertisseur de protocoles;

– l'étude du traitement des flots de paquets dans les pires cas de transfert à hauts débits.

Ces éléments nous permettent de solutionner notre problème de conversion à un niveau d'abstraction élevé, basé sur un ensemble de modèles.

3.2 MODÈLE D'ANALYSE DE LA PLATE-FORME SoC

La modélisation d'une plate-forme SoC consiste principalement à décrire de façon structurelle l'environnement nécessaire, ainsi que l'ensemble des outils et bibliothèques utiles à la réalisation du système. Une telle structure est envisageable dans le cas d'une conception de système dont la complexité dépend non seulement des applications supportées, mais surtout des besoins de conception rapide, de réutilisabilité des modules, de flexibilité architecturale et de possibles conceptions génériques des produits dérivants du concept fondamental du système.

Dans le cas de notre convertisseur de protocoles, nous voulons être en mesure de concevoir des produits dérivés à partir d'une définition générale de convertisseur de protocoles.

Rappelons que dans ce contexte, la plate-forme SoC décrit un environnement architectural créé pour faciliter la réutilisation des modules nécessaires à la conception d'un système sur une puce (SoC). Ce système peut être adapté à une classe d'applications. L'application privilégiée dans notre cas est la conversion des protocoles réseaux. Elle est similaire à plusieurs autres types de traitements dans les réseaux de

communication. On peut donc étendre cette classe à l'ensemble des traitements applicables aux réseaux de communication.

Le but est d'obtenir un système capable de synthétiser de façon générique des produits dérivés à partir d'un ensemble général de spécifications [8] [5]. Le diagramme bloc suivant présente le modèle de conception de la plate-forme.

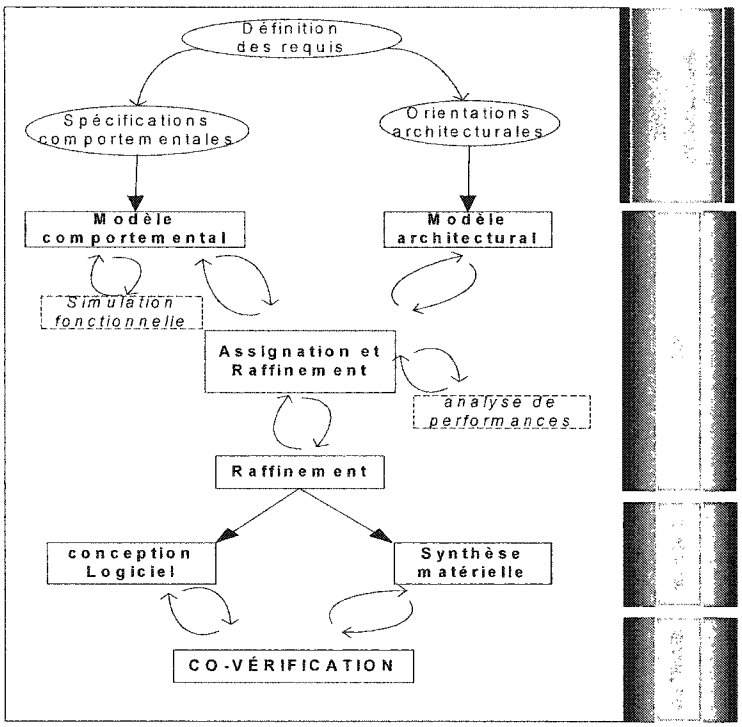

Figure 3-1 Étapes de conception haut niveau de la plate-forme SoC.
*(**Ellipse** : étapes manuelles; **rectangles gras** :étapes supportées par un outil; **rectangles pointillés** : résultats).*

La conception passe par des étapes de spécification, modélisation comportementale, modélisation architecturale, assignation/raffinement, synthèse logicielle, synthèse matérielle et enfin co-vérification. Dans la suite de ce chapitre, nous reprenons et élaborons sur chacune de ces étapes.

3.2.1 Spécification du système

Lors de la définition d'un système, on a deux types de spécifications:

- les spécifications non fonctionnelles qui décrivent les conditions de fonctionnement du système (consommation rendement etc…);
- les spécifications fonctionnelles dans lesquelles sont présentées les différentes fonctions recherchées, incluant les délais d'opérations.

Ainsi, la spécification haut niveau permet de décrire les éléments existants, d'en dégager les points forts, les insuffisances au niveau technique et de procéder au découpage du système par domaines d'activité [34]. Une fois les orientations techniques validées, on a recours aux outils de conception haut niveau.

3.2.2 Modélisation comportementale

C'est un modèle généralement séquentiel, géré par une horloge, qui permet d'attribuer des délais aux fonctions du système. Ces délais sont pré-évalués suivant les nombres de cycles requis pour chaque fonction. Les fonctions sont groupées en séries d'évènements, limitées par leurs temps d'exécution.

Cette modélisation peut se faire dans l'environnement VCC (Virtual Component Co-design) [5] en utilisant un langage de programmation de haut niveau. L'outil VCC décrit un modèle comportemental sous trois différentes formes dont la forme Blackbox, qui permet de modéliser le système en langage C++ et d'analyser les étapes d'exécution de la simulation fonctionnelle. La figure suivante représente le diagramme des classes du modèle comportemental du convertisseur de protocoles réseaux supportant les protocoles Firewire et Gigabit Ethernet.

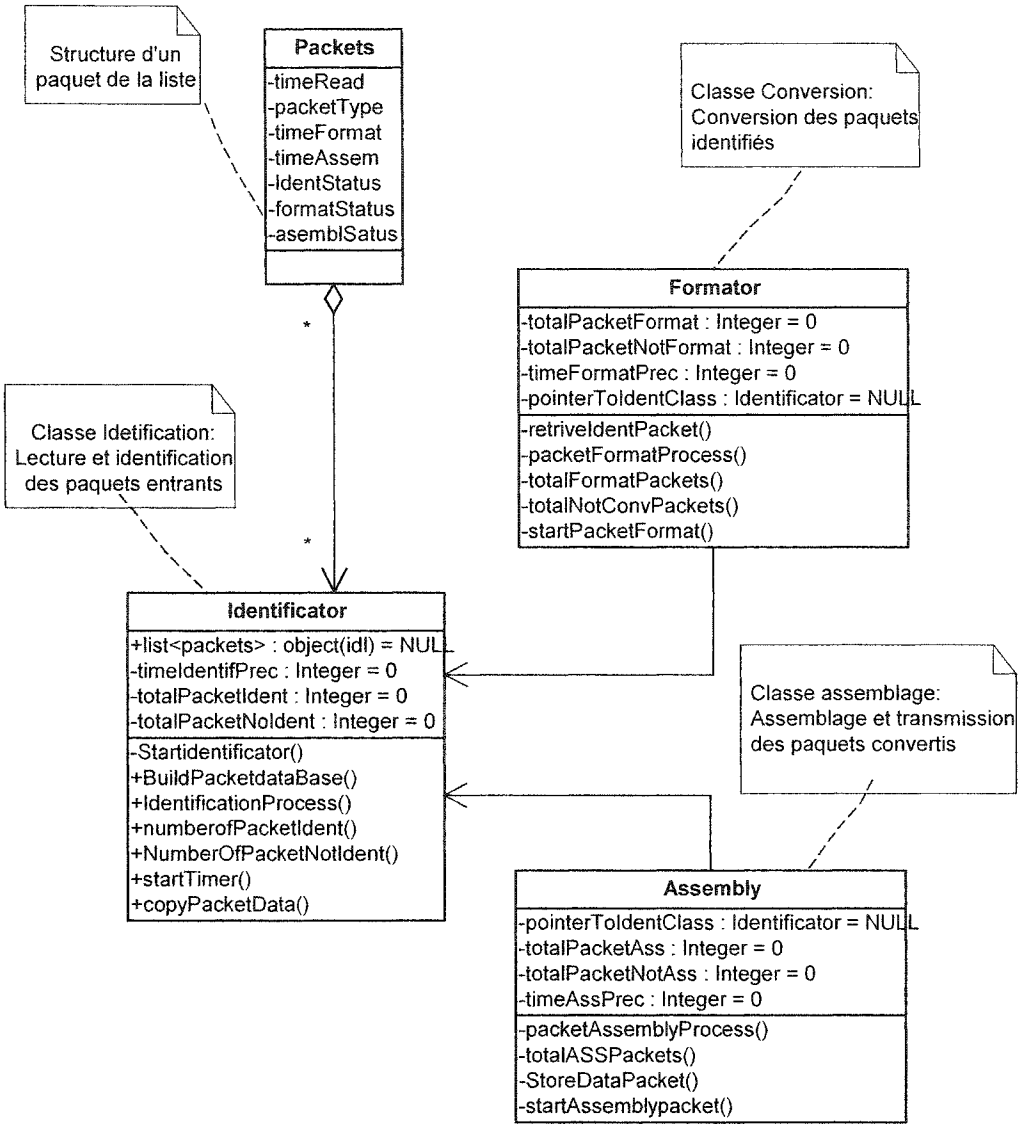

Figure 3-2 Diagramme de classes du modèle comportemental du système

Il a été développé en langage C++. Il contient trois classes:

– L'identification: Cette classe permet de lire les paquets entrants contenus dans le fichier d'entrée, d'identifier les adresses source et de destination de chaque paquet et de sauvegarder le contenu du paquet ainsi que les résultats d'identification dans la liste Packets.

– Formatage : Cette classe permet de lire les paquets identifiés de la liste Packets, de convertir les paquets identifiés et de sauvegarder les résultats du traitement dans la même liste Packets.

– L'assemblage : Elle lie les paquets convertis de la liste Packets, les assemble et transfère le résultat dans un fichier de sortie.

Un ensemble de vecteurs de test a été utilisé afin d'analyser les performances du comportement du système. La figure suivante présente une comparaison entre le comportement de l'architecture précédemment implémentée et la nouvelle architecture proposée en partie grâce à cette recherche. Cette nouvelle architecture est présentée à la figure 3-5. Les améliorations proposées sont de plusieurs natures.

Les modules de la version précédente ont été modifiés pour qu'ils soient facilement intégrables dans le nouveau système. Les principales améliorations apportées sur ces derniers ont été la modification des interfaces d'entrées et sorties, l'implémentation d'une méthode d'interconnexion point à point vers un seul bus reliant les divers interfaces et enfin le changement de la conception physique de chacun afin de pouvoir respecter les exigences de performances. Celles-ci étant nécessaires au bon fonctionnement de l'ensemble de cette nouvelle plate-forme.

La figure 3-3 présente la comparaison du taux de conversion entre l'arhitecture courante et la nouvelle architecture. On constate que le taux de conversion est de 100% dans cette nouvelle approche. L'amélioration du taux de conversion est justifiée par le fait que dans la nouvelle plate-forme, les temps de traitements et de transferts des paquets ont été considérablement réduits. Ceci est dû à l'augmentation des performances

des différents modules qui dans certains cas ont doublés leurs fréquences de fonctionnement par rapport à l'ancienne architecture. De plus, la nouvelle architecture supporte le traitement multitâche grâce entre autre, à son système d'interconnexion à bus partagés regroupant un ensemble de modules configurés à cet effet. Ces principaux éléments permettent à cette nouvelle plate-forme de garantir les traitements de flots de paquets même dans les cas extrêmes.

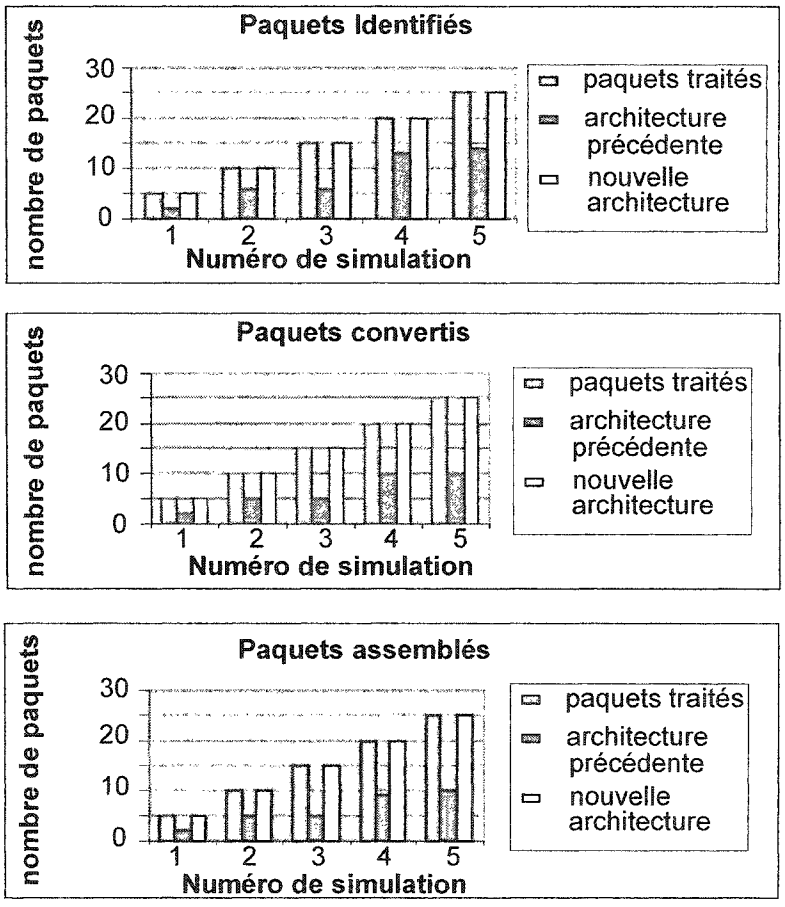

Figure 3-3 Comparaison du taux de conversion entre l'architecture courante et la nouvelle architecture.

3.2.3 **Modélisation architecturale**

Elle est limitée à la description des composants virtuels [5] à un niveau d'abstraction élevé. On définit simplement des modèles des performances recherchées pour chaque module de l'architecture. Une fois intégrés, ils constitueront l'architecture de notre système [8].

La complexité de la nouvelle architecture nécessite une modélisation en quatre étapes complémentaires:

- ressources coprocesseurs et processeurs;
- ressources de stockage de données;
- interconnexion des modules;
- ordonnancement du traitement.

Le modèle de performance de l'ensemble permettra d'analyser l'impact de l'architecture sur le comportement du système.

Les paramètres considérés pour l'analyse des modules de l'architecture sont :

Temps de traitement = temps d'exécution + temps de transfert + latence;

Latence = (nombre de cycles d'accès + cycles d'attentes) / (fréquence du module) ;

Largeur des mots de données = 32 bits pour tous nos modules.

Cette approche permet de sélectionner des IP (Intellectual Property) qui pourront être utilisés dans l'architecture lors de la synthèse, ou de définir les spécifications des modules à concevoir.

VCC qui permet une modélisation architecturale, n'a pu être utilisé, car sa bibliothèque de composants virtuels pour l'instant est très limitée. Son utilisation

nécessiterait une phase de conception des composants virtuels envisageables. Mais, cette étape supplémentaire aurait pour inconvénient d'augmenter le temps de conception du nouveau système. Pour y remédier, nous avons apporté des améliorations aux composants de la version précédente de l'architecture.

L'analyse faite sur l'architecture courante a permis d'apporter certaines améliorations sur ces différents modules tels que nous l'avons mentionné dans la section précédente. L'intégration de ces améliorations dans les fonctions matérielles du modèle comportemental du système, permet de déterminer les bandes passantes requises pour chacun de ces modules afin de garantir le traitement des flots de paquets. Le tableau suivant résume les bandes passantes requises des modules de la nouvelle architecture.

Tableau 3-1 Bandes passantes requises des divers modules.

Module	***Bande passante***
Processeur ARM	80 Mo/s
Identificateur de Paquets	160 Mo/s
Convertisseur d'adresses	160 Mo/s
Intégrité des données	160 Mo/s
Processeur de Formattage	160 Mo/s
Assembleur de paquets	160 Mo/s

La liste *Packets* spécifiée lors du modèle comportemental sera matérialisée par une mémoire SRAM qui devra posséder une bande passante au moins égale à la somme des bandes passantes requises des modules qui l'accèderont. Cette mémoire a été générée en utilisant le compilateur de mémoire VIRAGE [39]. Les principales caractéristiques de cette mémoire sont décrites au tableau 3-2.

Tableau 3-2 Principales spécifications de la mémoire du système

Type	Mémoire RAM synchrone
Nombre de port	1 port
Bus data-in / data-out	Séparés
Bus de données	32 bits
Bus d'adresses	14 bits
Bande passante	1280 Mo/s
Latence	Aucune
Taille	8 ko
Technologie	0.18 micron

Une interconnexion adéquate de l'ensemble de ces modules constituera la nouvelle architecture de notre convertisseur de protocoles. La structure d'interconnexions utilisée dans la plate-forme SoC proposée est basée sur les bus AMBA.

Le bus AMBA (*Advanced MicroController Bus Architecture*) est une architecture normalisée. Il est utilisé dans la conception de systèmes sur puce de haute performance. Il existe trois différents types d'architecture de bus AMBA [3]:

- le bus APB (Advanced Peripheral Bus): C'est un bus optimisé pour la réduction de consommation de puissance et pour la complexité d'interfaces. On peut l'utiliser conjointement avec deux autres types de bus AMBA à savoir AHB et ASB.
- le bus ASB (Advanced System Bus): c'est un bus utilisé dans la conception des microcontrôleurs embarqués de haute performance à 16 ou 32 bits de bus données. C'est une solution alternative qu'on utilise lorsque les performances en fréquence et les fonctions du type AHB ne sont pas requises. Le bus ASB a les mêmes connexions d'interfaces que le bus AHB.
- le bus AHB (Advanced High-Speed Bus): Il est utilisé pour des systèmes de haute performance nécessitant des fréquences d'horloge élevées qui offre différents modes de

transferts. Il supporte avec efficacité les connexions d'interfaces entre processeurs, coprocesseurs et mémoires.

La figure 3-4 présente le diagramme bloc du module d'interconnexion de la nouvelle plate-forme SoC du convertisseur de protocoles réseaux.

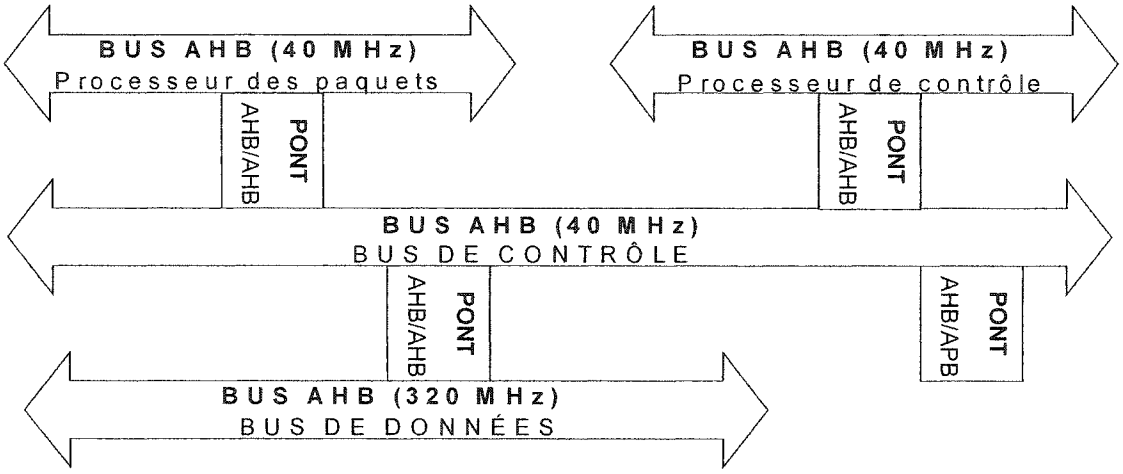

Figure 3-4 Structure d'interconnexion pour la nouvelle architecture

La structure proposée comprend quatre bus AHB (Amba High-speed Bus) liés par quatre ponts donc trois ponts AHB/AHB et un AHB/APB pour la connexion des périphériques [3].

Les deux bus AHB locaux sont inclus respectivement pour le processeur ARM7 de contrôle et le processeur ARM7 de traitement de paquets de l'architecture montrée plus loin. Un bus de contrôle AHB 40 MHz sert aux transferts des commandes. Enfin, le bus AHB 320 MHZ est utilisé pour l'accès à la mémoire principale par l'ensemble des modules du système.

3 .2.3.1 Nouvelle architecture du convertisseur de protocoles réseaux

L'architecture considérée est complexe à cause du nombre de modules qu'elle regroupe, mais d'autre part elle est relativement simple par son fonctionnement.

L'élément essentiel que nous présentons à la figure 3-5 est l'interconnexion des modules.

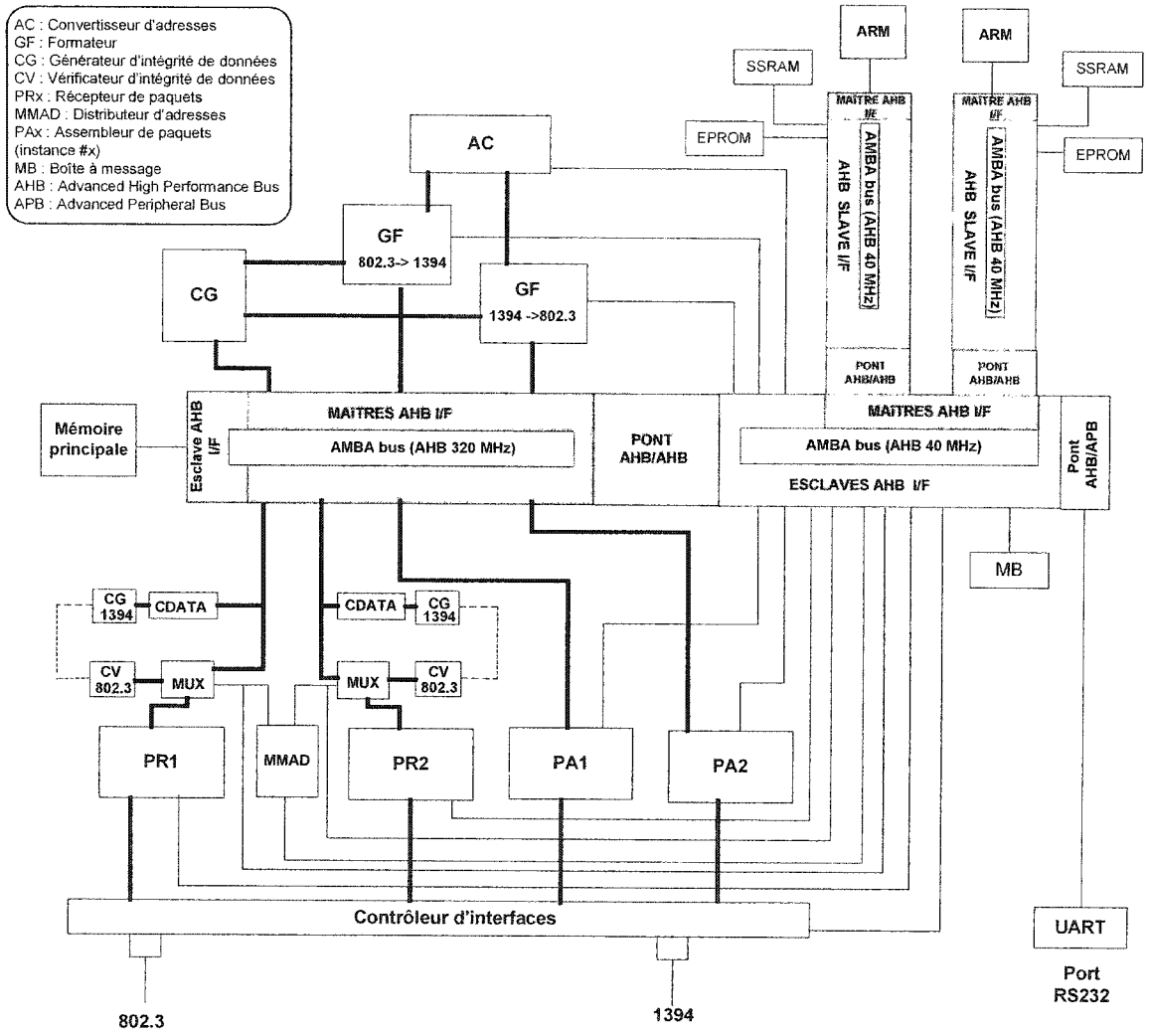

Figure 3-5 Schéma bloc de la nouvelle architecture du convertisseur de protocoles.

Tel que présenté dans la figure ci-dessous, le système d'interconnexions se compose de trois bus AMBA. Le premier, le bus AHB haute vitesse, est dédié aux transferts à hauts débits d'informations vidéo. Le second bus, le AHB basse vitesse, est réservé aux transferts d'informations de contrôle. Le troisième bus, le APB, relie des interfaces basse performance au système. Les bus AMBA AHB (Advanced High-performance Bus) et AMBA APB (Advanced Peripheral Bus) ont été choisis pour leur flexibilité et leur compatibilité avec les processeurs ARM.

Les interconnexions possibles dans cette architecture sont donc définies par un ensemble de bus AMBA qui sont liés par des ponts. Cette structure est caractérisée par la séparation du chemin de données et du flux de contrôle. Ceux–ci sont respectivement supportés par le bus AHB de 320 MHz et le bus AHB de 40 MHz. Le fonctionnement de cette structure adressable nécessite la contribution des deux processeurs ARM qui gèrent distinctement le contrôle du système et la mise à jour des champs d'adresses.

Cette architecture facilite l'intégration des modules quelques soient leurs choix d'assignation.

3.2.4 **Assignation**

C'est l'étape la plus importante du développement haut niveau. Elle consiste à partitionner les fonctions du modèle comportemental en deux groupes, matériel et logiciel, selon les avantages de fonctionnalités, de performances et de contraintes physiques [28].

L'analyse de l'implémentation de la version courante de l'architecture, sur la plate-forme Integrator A/P, a facilité l'étape de partitionnement. On a pu rapidement arrivé aux conclusions suivantes:

- L'identification de protocoles nécessite une implémentation en partie matérielle et logicielle;
- La conversion des données sera réalisée par un processeur spécialisé utilisant un jeu d'instructions adapté à l'application;
- L'assemblage des paquets doit être fait par un accélérateur matériel.

Ces résultats ont été considérés dans le partitionnement de la nouvelle plate-forme SoC de notre convertisseur de protocoles.

3.2.5 **Raffinement**

Le raffinement ou encore l'ajustement du système est fait après l'analyse des performances. Dans certains outils comme VCC, on utilise les modèles de performance de chaque module constituant l'architecture pour faire une évaluation du système construit. Ainsi, si les modules de notre architecture sont des IP existants dans l'une de ces bibliothèques, nous n'aurons plus besoin de définir le modèle de performance de ces modules. Cependant, VCC ne permet pas d'avoir la vraie fréquence fondamentale de chaque module. Pour les IP ou les modules développés par le concepteur, ceci ne devrait pas être un problème, car ces fréquences sont connues. Par contre, lorsqu'on importe un module développé par d'autres fabricants, ceci peut être un obstacle pour la conception si leurs spécifications des performances ne sont pas bien définies.

Une fois la validation des modèles du système terminée, ces modèles sont utilisés à l'étape de synthèse logicielle et matérielle dépendamment des assignations.

3.2.6 **Synthèse logicielle**

Cette opération permet de développer les applications logicielles du système. La synthèse logicielle est une opération qui ne sera pas traitée dans le cadre de notre travail, vue la complexité du système. Néanmoins, lors de l'étape d'intégration du système, nous développerons l'ensemble des applications nécessaires à la co-vérification de cette nouvelle plate-forme SoC.

3.2.7 **Synthèse matérielle.**

Il existe dans ce sens deux types de synthèse :

1) *la synthèse logique* : Elle permet de présenter la vue structurelle d'un circuit au niveau logique [34]. Dans ce contexte, il s'agit de manipuler des spécifications logiques afin de générer des interconnexions et des primitives logiques. Elle détermine la structure microscopique du circuit au niveau des portes logiques. L'étape finale, connue sous le nom de *mapping* technologique, permet de déterminer le type de porte de la bibliothèque à utiliser [6].

2) *La synthèse physique* : Elle permet de générer les motifs géométriques des masques de fabrication des circuits à partir d'une description logique du système [34].

La description du layout est la dernière étape de conception d'un circuit intégré. On l'utilise principalement pour la fabrication de la puce. Il est important que le dessin des masques respecte les règles de dessin du procédé de fabrication.

L'outil de synthèse matériel Synopsys permet de transformer une description comportementale du code en une description au niveau du transfert des registres (RTL ou Register Transfer Level) qui servira d'entrée pour l'outil de synthèse logique. Il permet de déterminer l'assignation des fonctions du circuit à des opérateurs appelés *ressources*, les interconnexions entre les différentes ressources, ainsi que le moment d'exécution de chaque opération dans un intervalle de temps, appelé pas de contrôle. Dans notre cas, la synthèse est effectuée en tenant compte de la surface et des fréquences requises.

Une synthèse matérielle pour la conception de notre plate-forme SoC est d'une importance particulière. Elle permet de définir un modèle structurel du chemin de données comme une interconnexion de ressources et un modèle de niveau logique de l'unité de contrôle. Celle-ci sera chargée de piloter les signaux du chemin de données en accord avec le résultat de l'ordonnancement (définition du moment d'exécution pour chaque opération de la description comportementale en tenant compte des dépendances fonctionnelles). Les principaux avantages de la synthèse matérielle sont les suivants :

- Réalisation rapide du système car la description comportementale est plus courte.
- Simulation rapide car la description est plus abstraite qu'avec une description RTL
- Réduction considérable du temps de mise sur le marché.

3.2.8 Co-vérification

La conception de systèmes mixtes (logiciel/matériel) est généralement associée à la conception des systèmes embarqués [35]. Un système embarqué sur une puce (SoC) est principalement constitué d'au moins un microprocesseur, de modules d'interconnexions,

d'interfaces physiques, d'une ou plusieurs mémoires et d'un ensemble de coprocesseurs pouvant être des blocs IP ou des modules spécifiquement conçus à cet effet [30]. L'une des étapes de la conception d'un tel système est la co-vérification.

Après l'étape de synthèse comportementale, dans laquelle on ajoute entre autre les notions de temps facilitant le partitionnement, le raffinement de spécifications se fait à l'aide des langages d'implémentations. Ainsi, les modules logiciels sont conçus en langage C++ et les modules matériels sont développés en VHDL. Au cours de cette étape, on procède à la vérification et à la validation du système.

L'outil Seamless nous permet d'effectuer cette vérification [25]. Dans l'annexe 1, la figure A1-1 présente l'intégration de la nouvelle plate-forme SoC dans l'environnement de vérification proposé. Les tests de vérification ont été élaborés suivant la méthodologie proposée par la Société Canadienne de Microélectronique [6].

À ce stade-ci du développement, les sous-modules sont développés et des bancs d'essais simples ont permis d'en vérifier les fonctions de base. La prochaine étape consiste à faire l'intégration des sous-modules en modules de niveau plus élevé. Le plan d'intégration est donné dans le chapitre 4, à la figure 4-1 « Schéma simplifié de la structure des modules », qui présentera plus en détail la structure d'interconnexion du système. Les deux modules de haut niveau qui doivent être construits sont le bus de contrôle (40MHz) et le bus de données (320 MHz).

3 .2.8.1 ***Communication entre domaines de bus***

Le pont AHB/AHB, au centre de la figure 3-5, permet de relier le bus AHB de contrôle 40 MHz et le bus AHB de données 320 MHz. La validation de ce module se fait

par une communication entre les domaines de bus. Pour concevoir ce pont, il suffit simplement de jumeler un synchroniseur 320/40 MHz avec un adaptateur 40MHz. Le pont joue un rôle d'esclave sur le bus 40MHz et un rôle de maître sur le bus 320 MHz. Ce qui permet à l'un des processeurs ARM, maître du bus de contrôle AHB 40 MHz, d'effectuer une requête sur l'esclave (la mémoire) du bus de données AHB 320MHz . Ainsi, Les requêtes de lecture et d'écritures en mémoire provenant du ARM circulent via le pont du bus 40MHZ vers le bus 320 MHz .

L'adressage du système se fait sur 32 bits. Ces bits sont répartis somme suit :

31 27 13 0

- Le bit 31 désigne l'espace d'adresse du bus 40 MHz.
- Le bit 27 désigne le pont vers le bus 320MHz.
- Les 14 derniers bits servent à préciser l'adresse mémoire.

L'un des exemples de tests permettant de valider le fonctionnement du pont AHB 40 MHz / AHB 320 MHz, entre les deux domaines de bus, consiste à effectuer une demande provenant du bus AHB 40 MHz vers le bus AHB 320MHz et d'envoyer une confirmation du traitement de ces demandes dans le sens contraire. Un banc d'essais simple a permis de valider son fonctionnement. Le caractère 'C' a été inscrit à l'adresse 0X88000050 et le caractère 'B' à l'adresse 0X88000040. La mémoire a ensuite été lue et les données reçues ont été envoyées à l'écran. La figure 3-6 montre le résultat de ce test. Nous constatons que le système fonctionne adéquatement.

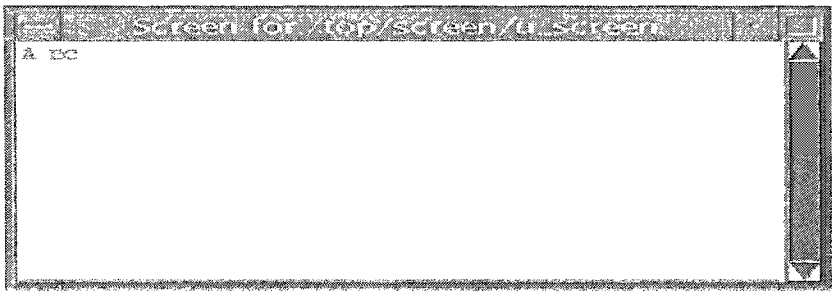

Figure 3-6 Communication via le pont AHB/AHB entre bus AHB 40 MHz et AHB 320 MHz.

Le premier caractère n'est pas pris en considération, car il s'affiche lors de l'initialisation du module Screen (écran) dans l'environnement de Seamless.

Nous avons effectué une série de tests pour vérifier la communication entre les deux domaines du bus AHB. La figure suivante présente un résultat plus détaillé de la communication entre le bus 40MHz et le bus 320MHz. Le transfert des données en lecture ou en écriture entre les deux domaines de bus AHB est mis en évidence.

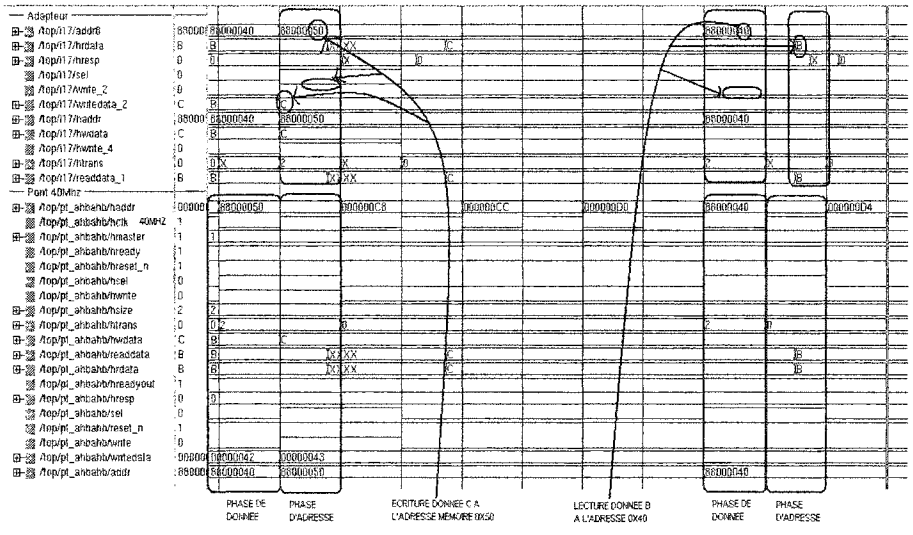

Figure 3-7 Chronogramme qui illustre le fonctionnement du pont AHB40MHz /AHB 320MHz

3 .2.8.2 Intégration des modules du domaine du bus de contrôle AHB 40MHz

L'objectif est d'utiliser un bus AHB à une vitesse de 40MHz permettant de gérer le processus de contrôle du système à l'aide des deux processeurs ARM qui y accèdent en tant que maîtres. Les autres modules du système sont esclaves à ce bus. Ils pourront donc réaliser leurs tâches selon les demandes émises par le processeur ARM de contrôle. Pour améliorer la performance du système, chaque processeur ARM aura sa propre mémoire et sera connecté au bus de contrôle via son propre bus AHB. Les deux bus AHB des processeurs ARM auront la même fréquence de fonctionnement que le bus de contrôle AHB, soit 40 MHz. Les modules esclaves du bus de contrôle seront connectés au bus via des interfaces (*wrapper*) qui assurera la synchronisation des signaux selon les différentes fréquences utilisées.

La structure interne du bus de contrôle est conforme à la norme AMBA AHB. L'une des vérifications de l'intégration de ce bus consiste à tester la communication entre les différents modules qui sont connectés.

Le test suivant consiste à vérifier le fonctionnement des sept modules esclaves connectés au bus de contrôle AHB 40 MHz. Il s'agit de demander à chacun des modules esclaves du bus de contrôle d'écrire respectivement une des lettres A à N en mémoire. Ensuite, le processeur ARM de contrôle relit l'information et l'affiche à l'écran. La figure 3-9 représente le résultat observé. Nous constatons que le bus de contrôle fonctionne correctement lors d'une transmission de flots de données.

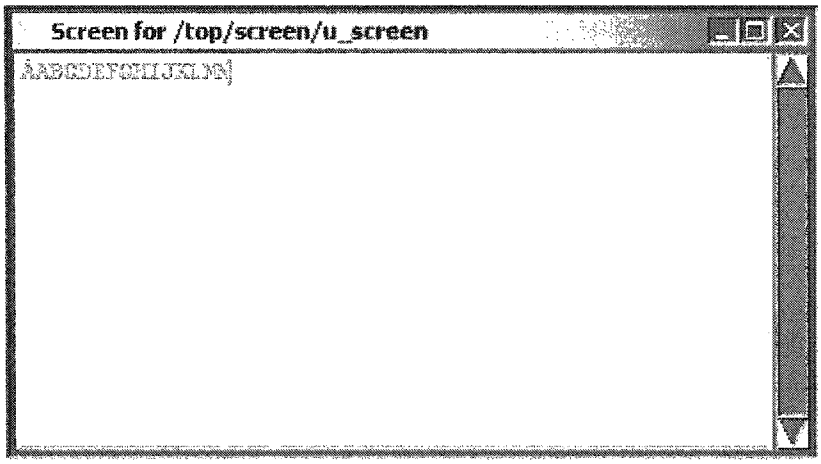

Figure 3-8 Exemple de communication des esclaves du bus de contrôle AHB 40 MHz

3 .2.8.3 ***Fonctionnement des différents modules***

Une méthodologie de test a été développée pour valider le fonctionnement des différents modules du système. Cette méthodologie ainsi que les différents bancs d'essais sont décrits en annexe 1.

Rappelons que les coprocesseurs de la nouvelle architecture sont répartis dans deux domaines de fréquence dont une partie fonctionne à 40 MHz et l'autre à 80 MHz. Le test de fonctionnalité des modules à été préalablement exécuté lors de leur conception. À l'étape de l'intégration de ceux-ci dans la nouvelle plate-forme, nous vérifions le fonctionnement des interfaces de communication de chacun (wrapper) d'une part et d'autre part le respect de la norme d'interconnexion des bus AHB.

Dans le processus de test, nous avons effectué des tests de lecture et d'écriture pour les différentes configurations du système. Certains exemples de tests mentionnés en annexe 1 tel que la lecture d'une donnée par un module maître (de fréquence 40 MHz ou 80 MHz) ou l'écriture d'une donnée par un module maître, nous montre que le système

fonctionne correctement selon la norme AHB AMBA. En effet, dans chaque cas la phase d'adresse permet de préciser le module à sélectionner ainsi que le type d'opération à effectuer. Par la suite, au prochain front d'horloge, la donnée est positionnée.

Le fonctionnement de l'arbitre de chacun des bus AHB a été vérifié. L'intégration de l'ensemble de ces bus à permis de valider le mécanisme d'arbitrage du système d'interconnexions de la nouvelle architecture.

Aussi, nous avons vérifié le fonctionnement du pont entre les deux domaines de bus et nous avons constaté qu'il respectait la synchronisation exigée par la norme AMBA

3 .2.8.4 ***Couverture du code des simulations***

Le *code de couverture* permet de connaître le pourcentage du code qui a été exécuté lors de la simulation par rapport au code non exécuté. La figure suivante présente le taux de couverture des modules intégrés dans cette étape.

Design Unit	Design Unit Type	Stmt Count	Stmt Hits	Stmt %	Stmt Graph
.and_2bit(untitled)	Architecture	1	1	100	
.bas_2bits(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
.basc1bit(untitled)	Architecture	6	6	100	
update_resp(untitled)	Architecture	107	107	100	
bas_2bits(untitled)	Architecture	6	6	100	
basc1bit(untitled)	Architecture	6	6	100	
bas_d(untitled)	Architecture	6	6	100	
decoder(untitled)	Architecture	1	1	100	
synchronisation(untitled)	Architecture	1	1	100	
tester5(tester)	Architecture	51	49	96	
bascule(untitled)	Architecture	4	4	100	
bascule1(untitled)	Architecture	4	4	100	
bascule(untitled)	Architecture	4	4	100	
bascule1(untitled)	Architecture	4	4	100	
bascule(untitled)	Architecture	4	4	100	
bascule1(untitled)	Architecture	4	4	100	

Figure 3-9 Taux de couverture du code

En général, tous les modules sont optimisés vu le pourcentage d'exécution de leurs codes. Dans certains cas, il a fallu supprimer certaines parties du code non applicables (ex : *when others* qui jouent le rôle d'exception). Le taux de couverture du système en entier ne peut être effectué à ce stade car les modules complémentaires ne sont pas encore intégrés.

3.3 CONCLUSION

Cette étude nous a permis de définir une nouvelle architecture de notre plate-forme SoC pour la conversion de protocoles réseaux. Pour y arriver, nous avons défini un modèle d'analyse à haut niveau d'abstraction. Ceci nous a conduit à définir un modèle comportemental et un modèle architectural du système. L'étape finale de conception consiste à valider la nouvelle architecture après synthèse composée de l'ensemble de ces modules.

L'élément nouveau de cette architecture est son système d'interconnexion des modules. Il s'agit d'une structure d'interconnexion à bus partagés. L'ensemble des bus utilisés a été conçu selon la norme AMBA. L'un de ces bus est le bus de données AHB 320 MHz. Sa conception nécessite une optimisation en fréquence et une synchronisation efficace avec les différents modules qui y sont connectés.

Ayant vérifié le fonctionnement de chaque module, nous avons pu procédé à des tests plus complexes afin de valider le fonctionnement de la communication du module d'interconnexion de cette nouvelle plate-forme. Dans le chapitre suivant, nous allons expliquer le concept d'interconnexion utilisé en insistant sur un élément particulier, à savoir le bus AHB de haute performance AHB à 320 MHz.

CHAPITRE 4

INTERCONNEXION D'UN BUS DE HAUTE PERFORMACE

Ce chapitre présente la conception du module d'interconnexion de la nouvelle architecture de la plate-forme SoC de notre convertisseur de protocoles. En première partie, nous allons regarder la structure d'interconnexion. Par la suite, nous présenterons les différentes étapes de la conception de son bus de haute performance, un AHB opérant à 320MHZ.

4.1 MODÈLE D'INTERCONNEXION

Le modèle d'interconnexion de la nouvelle architecture du convertisseur de protocoles réseaux utilise les bus AMBA du type APB et AHB. Comme nous l'avons déjà mentionné dans le chapitre précédent, la bus AHB est généralement utilisé pour les systèmes nécessitant des fréquences d'horloge élevées tels que les (co)processeurs et les mémoires. La conception du bus AHB à une fréquence de fonctionnement de plus de 320 MHz est celle que nous exposerons dans la suite du chapitre. Commençons par présenter la structure d'interconnexion des modules de la nouvelle plate-forme SoC au niveau système. Ensuite l'explication du mécanisme d'un simple transfert permettra de comprendre le fonctionnement du bus AHB.

4.1.1 Méthodologie de conception

Rappelons que ce projet consiste à mettre en place l'infrastructure de communication interne d'un système sur puce (*SoC*). La norme qui a été choisie est celle proposée par ARM, soit la norme du bus AMBA.

La méthode de conception employée est proposée par la CMC [6]. La première étape a permis d'identifier les fonctions du système et de réaliser une spécification exécutable sous la forme d'un modèle *temps réel.* Le modèle a été conçu à l'aide de processus concurrents (*capsules*), de sorte que le passage de la spécification exécutable au développement de bloc en HDL s'est fait sans difficulté.

4.1.2 Structure du système d'interconnexion

Les résultats de l'étape de conception du nouveau système sont présentés dans cette section. Les fonctions du système d'interconnexion sont définies et raffinées jusqu'à l'obtention de groupes de fonctions. Ces fonctions seront ensuite implantées dans des blocs lors de la prochaine étape de conception. L'organisation hiérarchique de cette structure est présentée à la figure 4-1.

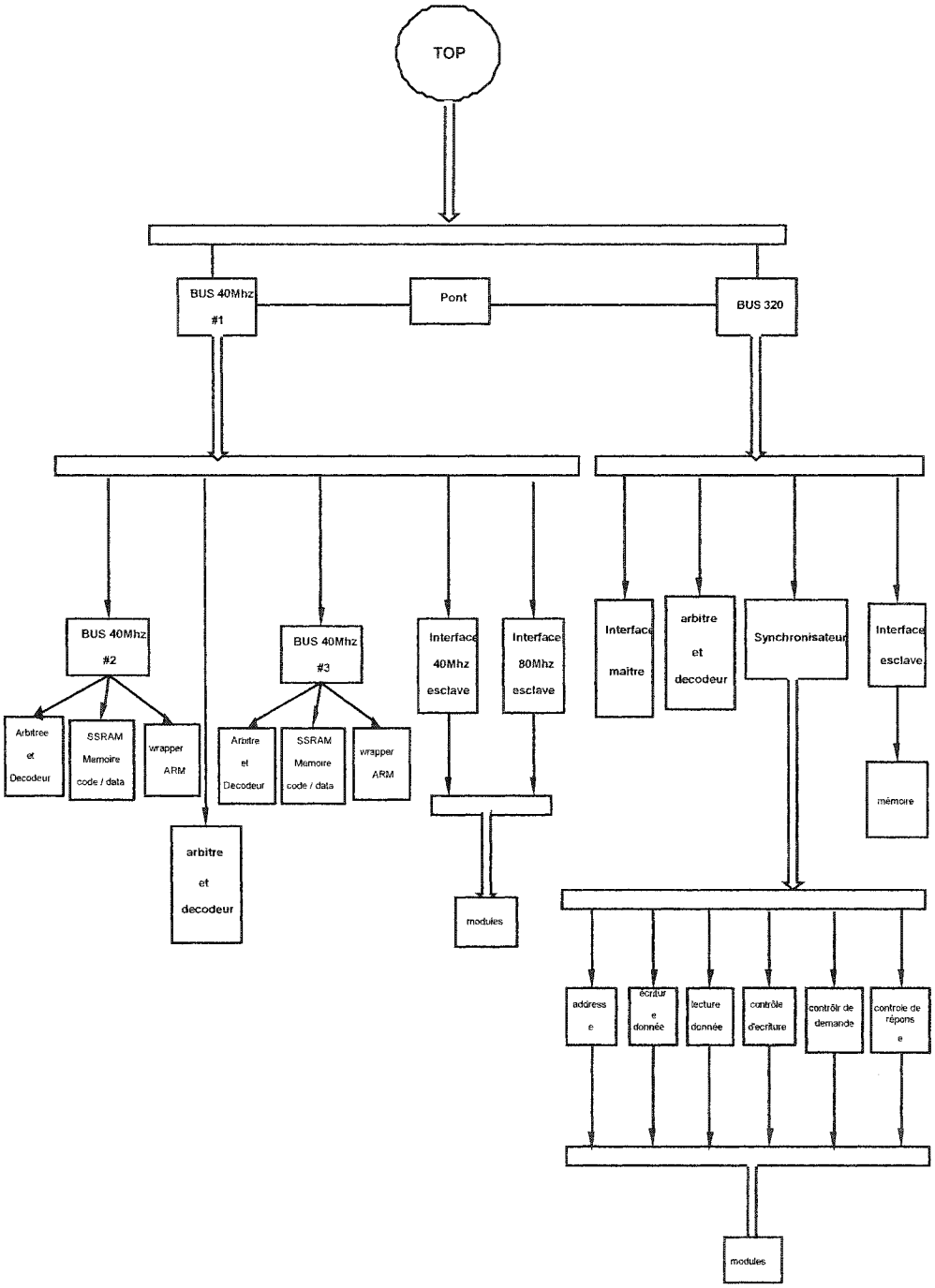

Figure 4-1 Schéma hiérarchique de la structure d'interconnexion

Le sommet de cette hiérarchie est constitué de deux domaines qui sont le bus de contrôle AHB 40 MHz et le bus de données AHB 320 MHz. Ceux-ci peuvent

communiquer à l'aide d'un pont AHB/AHB. Le reste des modules de l'architecture se retrouve dans la partie inférieure de cette structure selon les domaines auxquels ils sont applicables.

4 .1.2.1 ***Domaines de bus***

La séparation des interconnexions en différents domaines de bus est la première approche de haut niveau qui ait été effectuée. Après une analyse des besoins en performance des différentes parties du système, nous avons proposé l'utilisation de deux bus AHB donc un bus 40 Mhz et un autre de 320Mhz. Pour faire partie d'un domaine de bus, un bloc doit répondre à certaines exigences en matière d'interconnexion. Normalement, ces exigences sont atteintes grâce à une interface autour du bloc. La figure suivante illustre les domaines de bus du système.

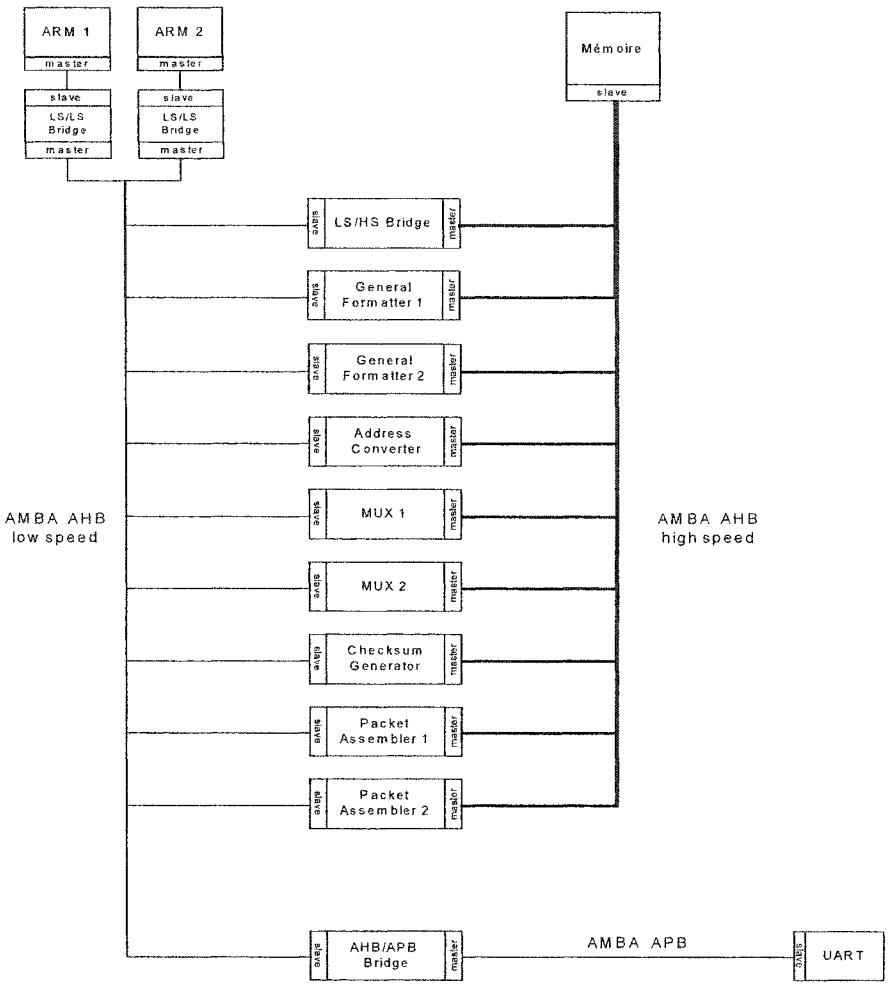

Figure 4-2 Domaines de bus du système

4.1.3 **Principe de communication d'un bus AHB**

La norme AHB définit un ensemble de règles de communication tel qu'il est décrit dans le document de spécification AMBA [3]. Ce document propose certains blocs afin de réaliser les fonctionnalités du système. Pour mieux comprendre son fonctionnement, il convient de présenter le chronogramme du déroulement d'un transfert sur un bus AHB. La figure 4-3 montre le chronogramme d'un simple transfert sur un bus de communication AHB.

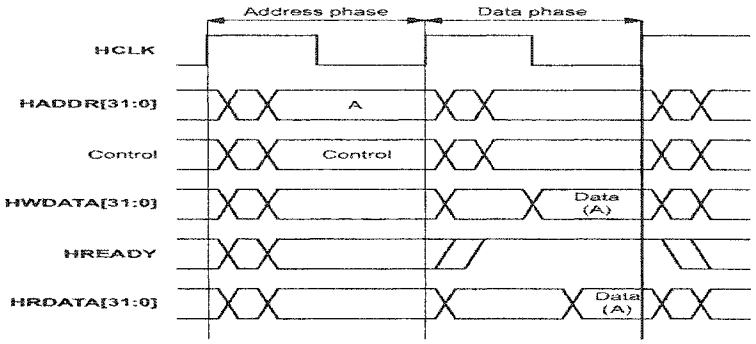

Figure 4-3 Chronogramme d'un transfert simple sur un bus AHB

Comme nous l'avons déjà mentionné, le transfert de l'adresse se fait un cycle avant le transfert de la donnée. L'élément important de ce chronogramme est la possibilité de *pipelinage* des accès. Pendant la phase « *data* » d'un certain accès, le prochain accès peut déjà entrer dans sa phase « *address* » et présenter son adresse sur le bus HADDR. On constate que la norme AMBA utilise les bus de données unidirectionnels donc un bus de lecture HWDATA et un bus d'écriture HRDATA. Le signal de contrôle HREADY est activer par l'esclave afin d'indiquer au maître actif qu'il peut procéder au transfert.

Le système proposé est composé de quatre bus AHB. La conception de ces bus est identique, sauf celle du bus 320 MHz qui impose plus de contraintes.

4.2 LE BUS AMBA AHB 320 MHz

Le bus AHB reprend les grands principes des bus pour cartes imprimées tels que le PCI ou le ISA, en les transposant sur un circuit SoC comme l'indique la figure 4.4.

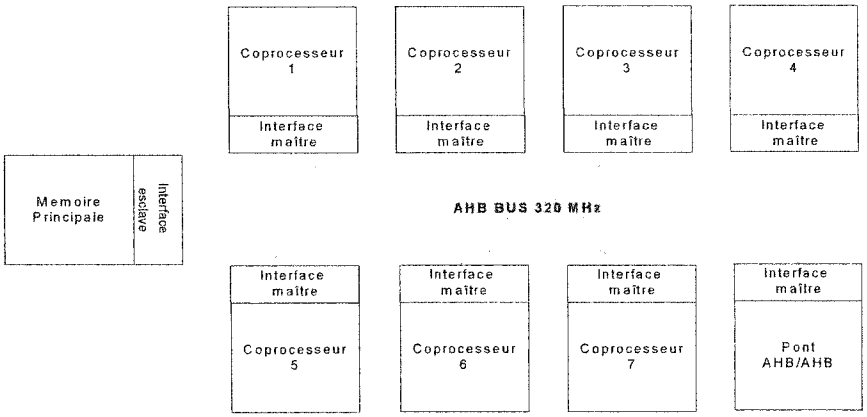

Figure 4-4 Schéma global du bus AHB 320 MHz

Ce bus se compose principalement des éléments suivants :

- un bus de données partagées, raccordé à tous les modules qui l'utilisent avec des chemins de données d'une largeur de 32 bits ;
- un bus d'adresses ;
- un arbitre gérant la communication de tous les modules maîtres du bus ;
- un décodeur, qui contrairement à l'arbitre, est relié à tous les éléments esclaves disponibles sur le bus.
- un ensemble de signaux de contrôle pour la gestion du système.

La figure 4-5 présente la structure interne du bus.

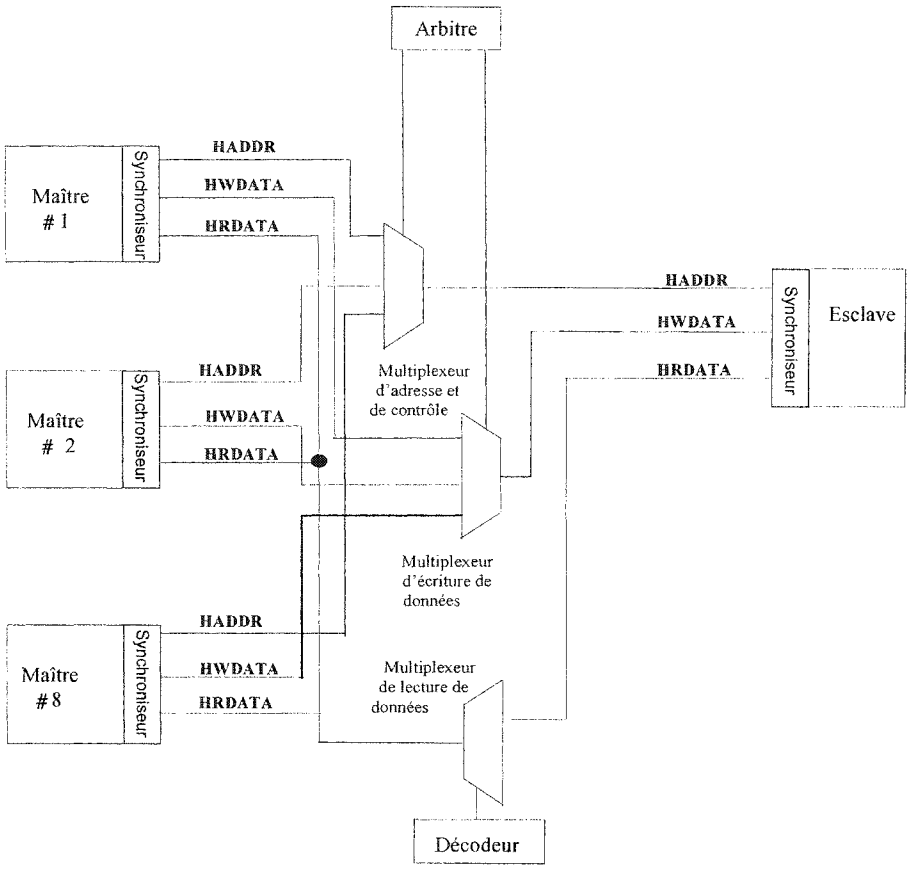

Figure 4-5 Architecture d'un bus AHB à huit maîtres et un esclave

Tous les éléments connectés sur le bus sont susceptibles de piloter ou d'échantillonner les lignes du bus de données, mais seuls certains d'entre eux peuvent piloter le bus d'adresses. Ces éléments sont appelés maître du bus. Les autres sont des esclaves.

4.2.1 **Module maître d'un bus AHB**

Il est capable d'amorcer les opérations de lecture et d'écriture en fournissant les adresses et les informations de contrôle. Un seul bus maître doit être actif à la fois. Le

maître AHB a l'interface de bus la plus complexe parmi les divers bus définis par la norme AMBA. Son diagramme d'interface est montré à la figure suivante :

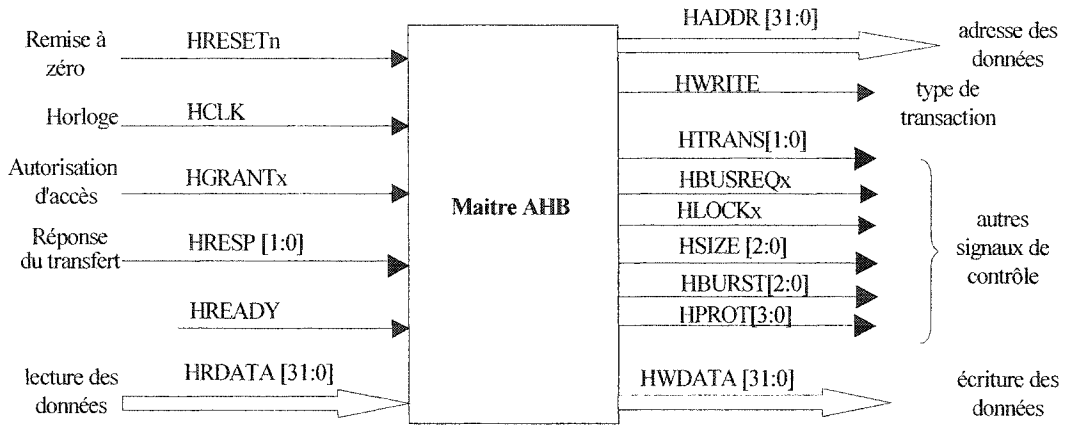

Figure 4-6 Architecture d'un maître AHB

Dans le cas du bus AHB 320 MHz, nous avons huit maîtres qui communiquent avec la mémoire principale grâce au système de gestion d'accès implémenté dans l'arbitre.

4.2.2 Interface maître du bus AHB 320 MHz

L'interface maître permet d'adapter un module quelconque en maître au bus AHB. Cette interface facilite la compatibilité entre un module et le bus AHB. Il assure également une bonne synchronisation des signaux durant un transfert. La figure suivante présente le diagramme bloc d'une interface maître.

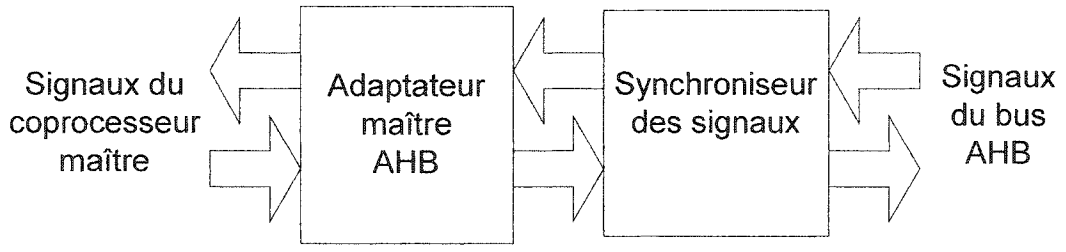

Figure 4-7 Diagramme bloc d'une interface maître du bus AHB 320 MHz

Il est très important de définir une bonne méthode de communication entre le module de synchronisation de signaux et l'interface, afin de minimiser les délais de transfert. Aussi, pour optimiser la communication entre les modules, nous avons restreint le nombre de signaux de contrôle. La figure suivante représente le chronogramme utilisé d'une communication à travers ce bus AHB.

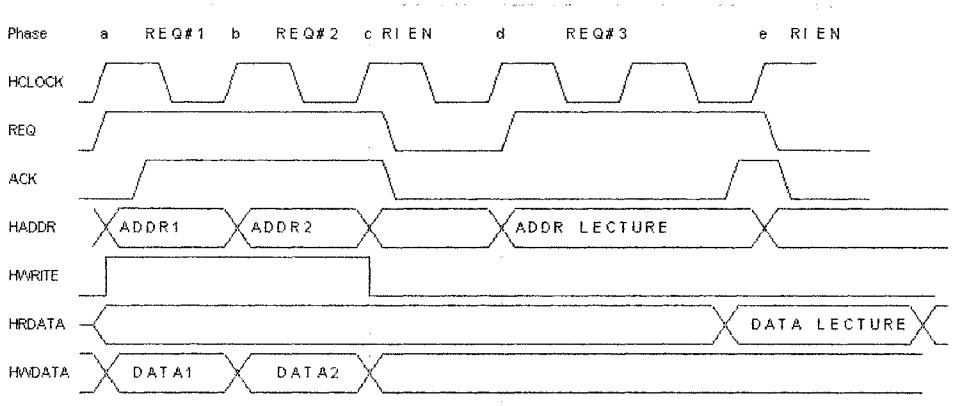

Figure 4-8 Chronogramme de lecture/écriture vue d'un maître du bus AHB 320 MHz

Comme l'indique le chronogramme, l'étiquette phase représente les requêtes émises sur le bus. Ainsi, les phases REQ#1 et REQ#2 sont des demandes d'écritures provenant d'un module maître vers la mémoire. Tandis que la phase REQ#3 est une demande de lecture. Lors d'une remise à zéro du signal REQ, le signal ACK revient automatiquement à zéro. On devra donc maintenir REQ actif tant que la demande n'est pas traitée. Comme l'adresse (HADDR), la donnée (HRDATA/HWDATA) et le signal de contrôle lecture/écriture (hwrite) sont mémorisés dans le wrapper, il n'est pas nécessaire de le maintenir dans le module. En conformité avec la norme AMBA, nous

utilisons le bit le plus significatif de HTRANS pour représenter le signal REQ et le bit le plus significatif de HRESP pour indiquer le signal ACK.

4.2.2.1 Les entrées/sorties nécessaires d'un module maître du bus AHB 320 MHz

L'utilisation d'une interface entre module et bus rend la conception des modules indépendante à la norme AMBA. Pour simplifier l'intégration des modules une fois conçus, nous avons défini les signaux d'entrées/sorties nécessaires que doivent posséder un module pour qu'il puisse être connecté en tant que maître au bus AHB 320 MHz. Le tableau suivant représente ces signaux.

Tableau 4-1 Signaux E/S nécessaires d'un module maître du bus AHB 320 MHz

Nom du signal	**Description**
HCLKx	Signal d'horloge du module x. Utilisé pour la synchronisation entre le module et le bus
REQ	Indique une demande d'accès au bus.
ACK	Signal de réponse au traitement d'une demande d'accès au bus
HWRITE	Signal indique le type de transaction à effectuer soit : lecture ou écriture
HADDR(31 :0)	Bus d'adresse. Dans ce bus, les 14 bits les moins significatifs indiquent l'adresse mémoire
HRDATA(31 :0)	Bus de données pour effectuer une lecture
HWDATA(31 :0)	Bus de données pour une écriture

Rappelons que ce bus AHB 320 MHz permet une connexion de huit modules maîtres et d'un module esclave. La fréquence de fonctionnement des modules maîtres est de 40 MHz ou 80 MHz. Les autres signaux non utilisés de la norme AHB d'un module maître sont optionnels. Comme le montre le chronogramme de la figure 4-10, un module maître de fréquence 40 MHZ ou 80 MHZ est capable d'envoyer des données en rafale via le bus AHB 320 MHz.

4.2.3 Module esclave d'un bus AHB

L'esclave répond aux requêtes d'écriture et de lecture suivant l'emplacement de l'espace adresse. Il renvoie la réponse au maître actif par un signal de succès, d'échec ou d'attente du transfert de données. Il utilise le signal de sélection *HSELx* provenant du décodeur pour savoir à quel moment il doit répondre à un transfert. Tous les autres signaux requis pour le transfert, qui fournissent les informations de contrôle et l'adresse, sont genérées par le maître actif.

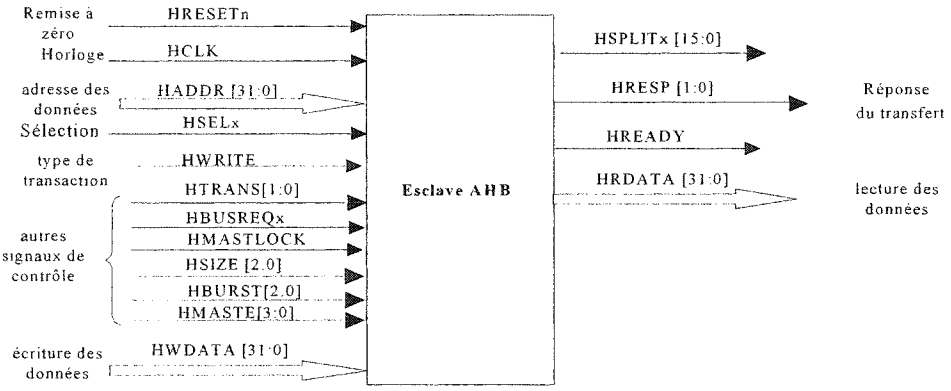

Figure 4-9 Architecture d'un esclave AHB

Dans le cas du bus AHB 320 MHZ, nous avons un esclave, qui est la mémoire principale du système, et huit maîtres qui sont respectivement les sept coprocesseurs du système et le pont reliant le bus AHB 320 MHz au bus AHB du système de contrôle. Ces modules ont été conçus dans l'optique de pouvoir être facilement adaptables à la norme AMBA. Ainsi, chaque module est relié au bus par une interface qui assurera la synchronisation et l'adaptation des signaux.

4.2.4 Interface esclave du bus AHB 320 MHz

L'interface esclave permet d'adapter la mémoire du système pour qu'elle se comporte comme un esclave du bus AHB 320MHz. Sa structure est la même que celle de l'interface maître. La seule différence est au niveau des signaux d'un esclave et de leurs fonctionnalités par rapport aux signaux d'un maître.

Pour la même raison d'intégration, nous avons défini entre le module esclave et le bus AHB 320 Mhz une interface différente. La figure suivante représente le chronogramme utilisé pour la conception de cette interface.

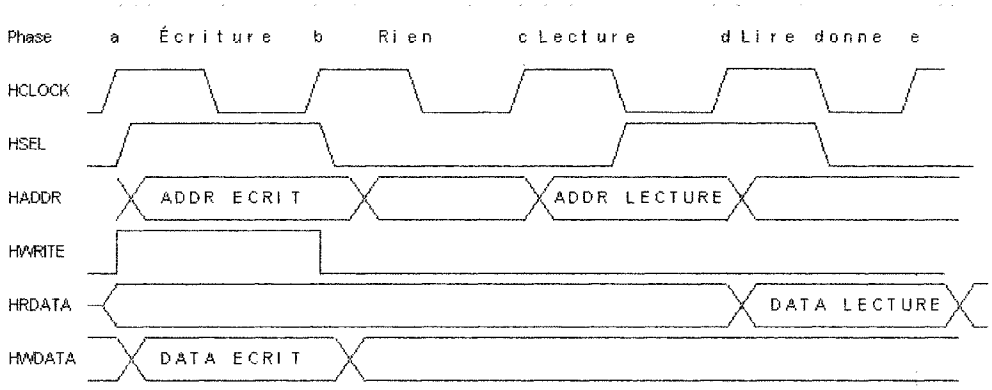

Figure 4-10 Chronogramme de l'interface esclave du bus AHB 320 MHz

Comme nous l'avons mentionné plus tôt, le seul esclave connecté sur le bus est la mémoire principale. Les huit autres modules connectés au bus sont des maîtres. En effet, le pont AHB/AHB est maître du côté du bus AHB 320 MHz et esclave du côté du bus AHB 40 MHz. Ceci permet au processeur ARM de contrôle, qui est maître par défaut sur le bus de contrôle AHB 40 MHz, d'interroger l'ensemble des modules du système et de procéder ainsi à une lecture ou une écriture.

Dans le cas d'une lecture, le ARM adresse le registre du module en utilisant le bus d'adresse HADDR pour identifier son emplacement. D'après les spécifications de la norme AMBA, le format de bits du bus d'adressage HADDR est libre au concepteur. Seule ses huit bits les plus significatifs (31 à 24) sont réservés à l'identification du bus.

La phase C du chronogramme de la figure 4-10 est une demande de lecture. Dans cette période, le signal de type de transfert HWRITE est égal à zéro. Un cycle plus tard, la donnée est disponible au 'wrapper'. L'esclave est supposé répondre au cycle suivant. Si ce n'est pas le cas, le signal HREADY est remis à zéro. Dans la phase D, l'esclave échantillonne les contrôles et l'adresse pour répondre à la requête. Ce qui permettra à l'interface d'échantillonner la donnée sur la phase E. Les autres signaux de contrôle du bus AHB ne sont pas utilisés dans le but de simplifier le mécanisme de transfert et de réduire ainsi les délais de traitement.

Dans le cas d'une écriture, l'adresse d'écriture est mise sur le bus HADDR et le signal HWRITE est égal à un. La donnée est simultanément placée sur le bus HWDATA. Au cycle suivant, dans ce cas le cycle B, la donnée est échantillonnée par l'esclave concerné.

4.2.4.1 ***Entrées/ sorties nécessaires de la mémoire***

Nous avons défini un ensemble de signaux nécessaires que doit posséder la mémoire du système. Le tableau suivant représente ces signaux.

Tableau 4-2 Signaux E/S nécessaires de la mémoire

NOM	DESCRIPTION
ADDRi[13:0]	Le bus d'adresse.
Di[31 :0]	Le bus de donnée
ME	Active ou désactive la mémoire
WE	Niveau haut pour écrire sinon lecture
OE	Met les sorties de la mémoire en haute impédance
CLK	L'horloge
Qi[31 :0]	Le bus de données en lecture

4.2.5 Synchroniseurs d'interfaces

Les modules conçus pour la plate-forme SoC du convertisseur de protocoles fonctionnent à des fréquences différentes. L'établissement d'une interconnexion entre ces modules exige une certaine synchronisation pour garantir le bon fonctionnement du système. De même, pour éviter la perte des données et la création de goulots d'étranglement dans le transfert d'informations, nous avons défini des principes de synchronisations conformément à la norme AMBA.

Dans la figure suivante, nous présentons un aperçu général des synchroniseurs utilisés entre les modules maîtres et le bus AHB 320 MHz.

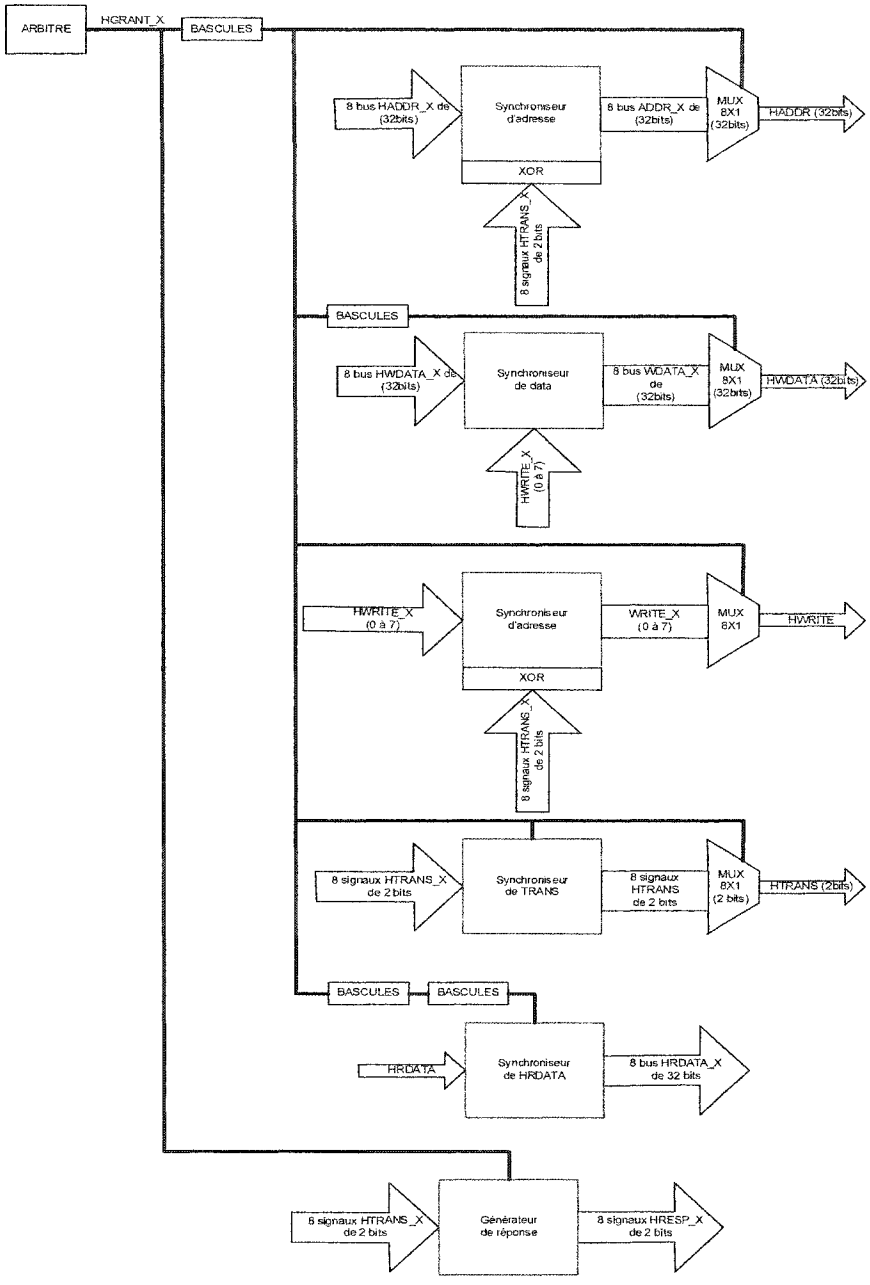

Figure 4-11 Schéma bloc de l'interface des maîtres du bus 320 MHz.

On peut voir qu'il existe des synchroniseurs sur tous les signaux E/S des modules maîtres à savoir: les bus d'adresses (HADDR), de données (HRDATA, HWDATA) et

les signaux de contrôles (HWRITE, HTRANS, HRESP). Cependant, ces synchroniseurs ont une architecture qui diffère selon les signaux.

4 .2.5.1 Caractéristiques des synchroniseurs

Pour contourner le problème introduit par les différentes fréquences d'horloges entre les modules et le bus, il est indispensable d'inclure des modules de synchronisation entre les signaux des modules et ceux du bus. Le but d'un synchroniseur est d'assurer une transmission synchrone entre les signaux provenant des domaines de fréquence différents. Dans la conception de ce bus, nous avons développé une série de synchroniseurs selon le type de signal à adapter.

La structure de base d'un synchroniseur de cette architecture est représentée à la figure 4-12. Il s'agit du synchroniseur du bus d'adresses HADDR .

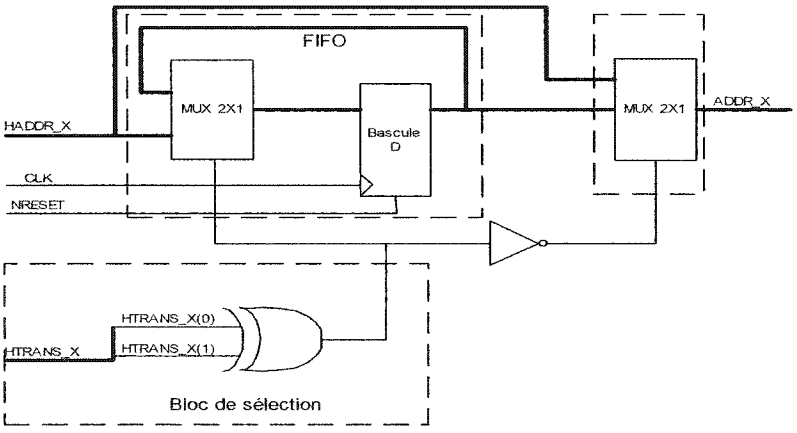

Figure 4-12 Schéma logique du synchroniseur d'adresses.

On veut transmettre l'adresse HADDR_x, provenant du module x vers le bus AHB. Le tampon (FIFO) est construit par le premier multiplexeur et la bascule. Le signal de demande HTRANS_x provenant du module x sert à sélectionner la valeur HADDR_x

échantillonnée à la fréquence du bus AHB 320MHz. Cette valeur est présentée à l'entrée du bus (ADDR_x) et change d'état au front montant de l'horloge du bus.

*4 .2.5.2 **Différences entre synchroniseurs***

Comme le montre la figure 4.12, la majorité des synchroniseurs ont été conçus sur une même architecture composé d'un FIFO (premier multiplexeur et bascule), d'un multiplexeur (deuxième multiplexeur) et d'un bloc de sélection (XOR). Cependant le bloc de sélection diffère selon le type de signal à synchroniser. Le tableau suivant résume les différences des blocs de sélection de ces synchroniseurs.

Tableau 4-3 Sommaire des différents modules de sélection des synchroniseurs

Signal a synchroniser	Description du signal	Bloc de sélection
HADD_x	Bus d'adresses provenant du maître x	Porte logique XOR avec les deux bits du signal HTRAN_x
HWDATA_x	Bus de données d'écriture provenant du maître x	Signal HWRITE_x
HRDATA	Bus de données provenant de l'esclave (mémoire principale)	HGRANT_x décalé de trois coups d'horloge du bus AHB (320 MHz)
HWRITE_x	Signal de contrôle d'écriture provenant du maître x	Porte logique XOR avec les deux bits du signal HTRAN_x
HTRANS_x	Signal de contrôle de type de transfert provenant du maître x	HGRANT_x sélection du maître x

*4 .2.5.3 **Autres synchroniseurs utilisés.***

Tel que montré à la figure 4-13, certains signaux nécessitaient une structure de synchronisation particulière.

1) Synchroniseur de réponse : HRESP

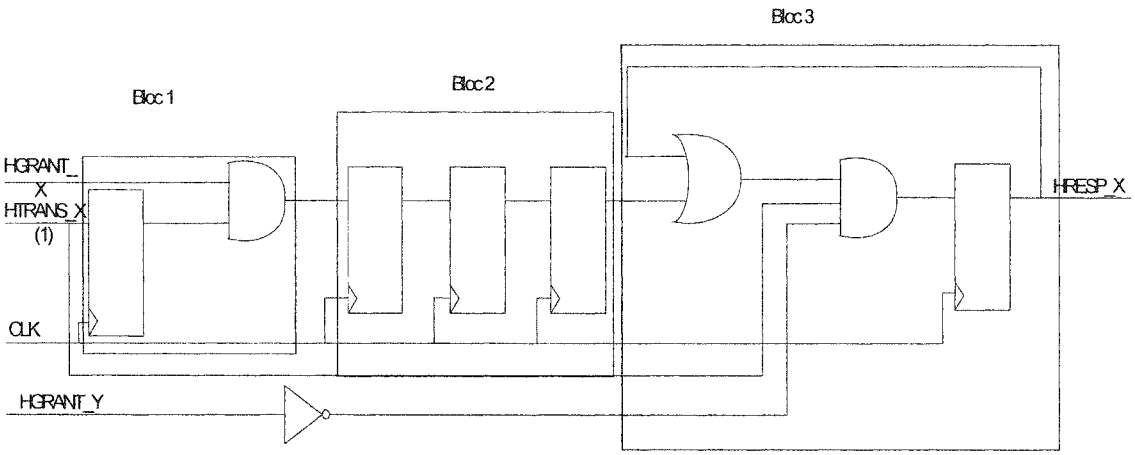

Figure 4-13 Schéma logique du synchroniseur de réponse

Le bloc 1 génère une impulsion de synchronisation lorsque le maître est sélectionné par l'arbitre. La bascule permet une synchronisation entre les domaines d'horloges du maître actif et du bus. Ceci permet de synchroniser la demande faite par le maître à la fréquence du bus.

Le bloc 2 crée le délai de réponse nécessaire au module pour compléter sa transaction de lecture ou d'écriture. Dans le cas de notre bus, il faut un minimum de quatre cycles avant de générer une réponse. Ce délai de réponse correspond au temps de transfert d'une lecture simple de la mémoire vers le module maître actif, plus un cycle de réponse nécessaire pour la mémoire.

Le bloc 3 mémorise la réponse HRESP_X . HRESP_X reste à '1' durant la requête du maître et revient à zéro lorsque la requête (HTRANS_X(1) = '0') du maître n'est plus présente. Le système mémorisera la réponse pendant un cycle d'horloge du module maître actif, après quoi elle est annulée.

2) Synchroniseur de l'interface esclave : mémoire

Le seul esclave connecté au bus AHB 320 MHz est une mémoire qui fonctionne à la même fréquence que le bus. Donc la synchronisation qui existe à ce niveau est le rattrapage du bus d'adresse qui est décalé d'un cycle par rapport à la donnée, tel qu'indiqué dans la norme AMBA. La figure suivante illustre le module synchroniseur de la mémoire.

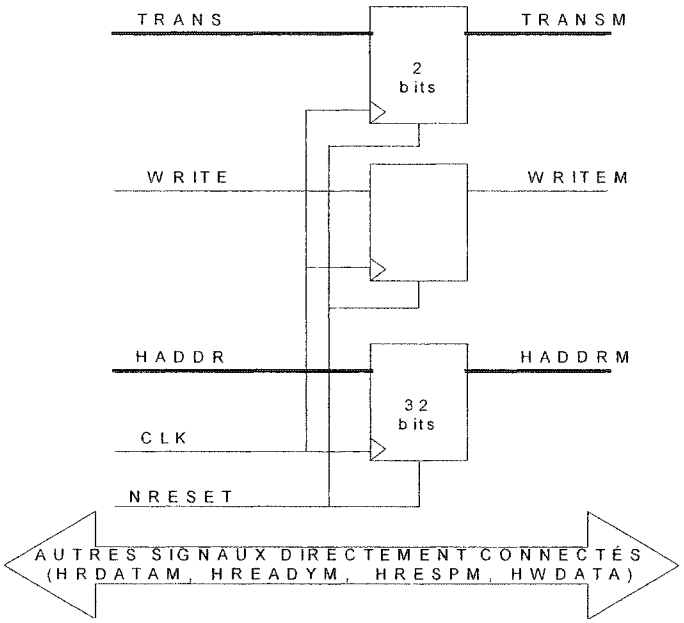

Figure 4-14 Diagramme bloc du synchroniseur de l'interface mémoire

4.2.6 Le décodeur du bus AHB 320 MHz

Le décodeur d'un bus AHB permet de sélectionner l'esclave en demande. Dans notre cas, nous avons un seul esclave qui est la mémoire principale. Le décodeur de ce bus sera une simple connexion entre le bus et le signal de sélection HSEL de l'interface mémoire.

4.2.7 L'arbitre du bus AHB 320 MHz

L'arbitre du bus AHB est le seul module de la norme AMBA dont la réalisation est libre au concepteur. En effet, sa structure est définie par un ensemble de signaux de contrôle permettant d'allouer des périodes d'accès aux différents maîtres du système. Ainsi, l'arbitre décide sur l'acceptation ou non de l'accès et du type de transfert supporté par le bus. La structure de la figure 4-15 représente le diagramme bloc d'un arbitre AHB.

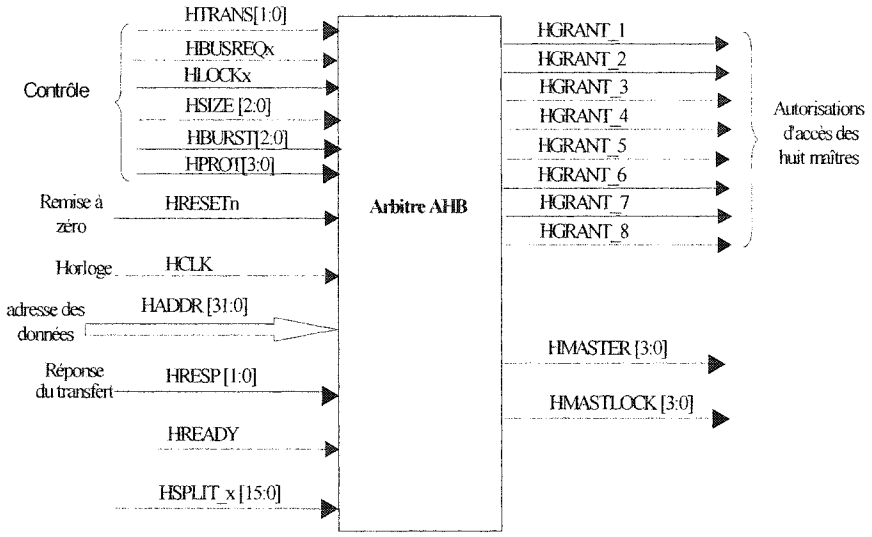

Figure 4-15 Architecture d'un arbitre de bus AHB à huit maîtres

4.2.7.1 Mécanisme d'arbitrage du bus AHB 320MHz

L'arbitre s'assure qu'il n'y a qu'un seul bus maître à la fois alloué au transfert de données. Même si le protocole d'arbitrage est fixe, n'importe quel algorithme d'arbitrage d'accès peut être implémenté dépendamment des exigences de l'application.

Dans le cas de notre application, nous avons implémenté un algorithme d'arbitrage qui permet un nouvel accès au bus à chaque cycle de son horloge. Ainsi, à chaque nouveau cycle d'horloge du bus, un nouveau maître a accès au bus. Ainsi, pour minimiser les demandes d'accès manquées, nous avons prédéfini l'accès au bus en fonction de la demande de chaque module. Le diagramme bloc suivant montre le mécanisme d'arbitrage implémenté pour gérer le bus AHB 320 MHZ.

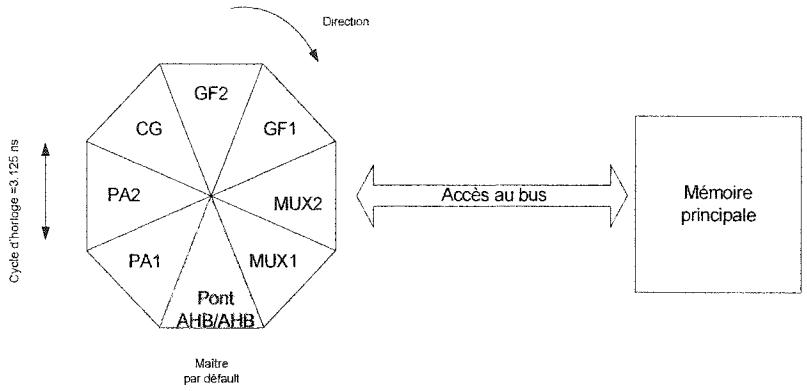

Figure 4-16 Mécanisme d'arbitrage du bus AHB 320MHz

L'arbitre alloue les intervalles d'accès de manière prédéterminée et selon un ordonnancement bien défini. Les huit maîtres connectés sur le bus peuvent accéder à la mémoire durant un cycle d'horloge et suivant l'ordre défini comme l'indique la figure 4-16. Le maître par défaut du bus est le pont AHB/AHB.

Une manière simple d'implémenter cet arbitrage est l'utilisation d'un registre à décalage comme l'indique le schéma logique 4-17.

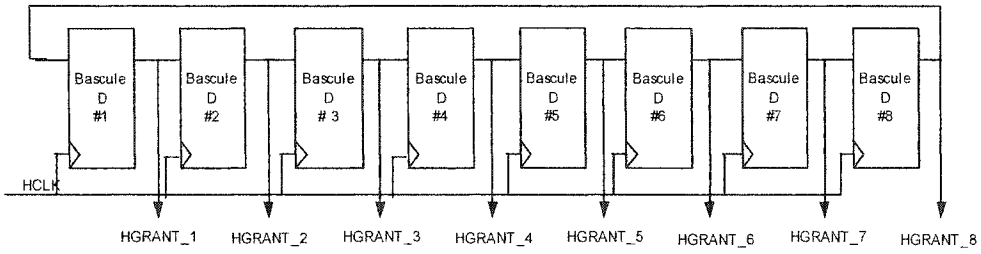

Figure 4-17 Schéma logique de l'arbitre du bus 320MHz.

Cependant notons que cette structure peut aussi être modifiée afin de rendre l'arbitre programmable en fonction des applications utilisées. Cette approche est utilisée dans la plate-forme μPlat [22] que nous avons présenté au chapitre 2.

4 .2.7.2 ***Simplification de l'arbitre 320 MHz***

Le mécanisme d'arbitrage utilisé doit assurer que juste un maître aura accès au bus en tout temps. Comme algorithme adéquat, la méthode de ''Round Robin scheduling'' a été implémentée. Les détails du processus de décision sont comme suit :

L'arbitre accorde la priorité au maître prédéfini pour accéder au bus. Au coup d'horloge suivant, le maître courant perd son accès et ce dernier est attribué au maître suivant de la file d'attente. Si plusieurs maîtres requièrent l'accès au même moment, la priorité est toujours attribuée selon la file d'accès prédéfinie. Un maître ne peut bloquer l'accès au bus. Si un maître ne demande pas l'accès au bus durant sa période réservée, cet accès lui sera néanmoins accordé suivant la politique prédéfinie. Les transferts en rafale sont supportés par le fait que le bus fonctionne à une fréquence de 320 MHz, alors que les modules présentent leurs requêtes à des fréquences de 40 MHz ou de 80 MHz. Chaque module pourra mémoriser sa requête d'accès dans son interface tant qu'il n'a

pas la priorité d'accès au bus. Une requête abandonnée ne sera pas considérée au moment de la décision.

Les simplifications effectuées lors de la conception de l'arbitre sont les suivantes :

- Les transactions *SPLIT* (accès éclatés) ne sont pas supportées;
- Le verrouillage du bus (*lock*) n'est pas supporté;
- L'arbitre peut contrôler jusqu'à huit maîtres;
- L'arbitre ne tient pas compte du type de transfert (rafale, *split* ...);
- L'incrément des rafales n'est pas supporté;
- La fin prématurée d'une rafale n'est pas supportée.

4.2.7.3 ***Règle d'arbitrage du bus AHB 320MHz***

La figure suivante illustre l'idée générale du traitement de requête de l'arbitre.

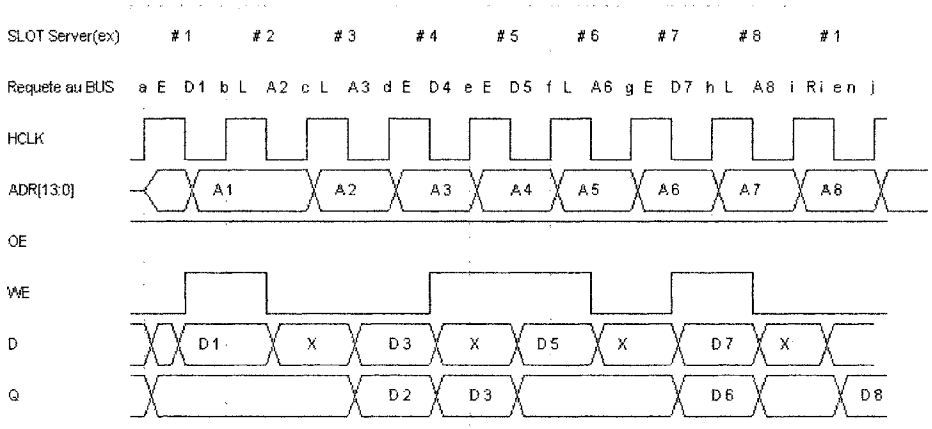

Figure 4-18 Chronogramme de partage de la ressource mémoire.

Pour résoudre le problème de compétition d'accès au bus entre maîtres, ceux-ci doivent généralement soumettre une requête à l'arbitre qui en élira un unique. Dans le cas de notre bus, nous prédéfinissons l'accès au bus afin d'éliminer le délai de traitement

des requêtes. Les modules n'ont plus besoin d'émettre une requête à l'arbitre car ils ont déjà des temps d'accès réservés. Ainsi, le module élu disposera du bus pendant le temps qui lui a été réservé et le décodeur sera chargé de déterminer les poids forts de l'adresse et de sélectionner l'esclave qui doit répondre à la transaction. Cette centralisation de l'arbitrage et du décodage est une caractéristique spécifique qui distingue les bus intégrés des bus périphériques.

4.3 TESTS ET RÉSULTATS

Cette section présente les méthodes de test choisies, ainsi que le plan de vérification des bus AMBA. La vérification du système d'interconnexion poursuit trois buts:

- Vérifier la conformité du modèle avec la spécification AMBA;
- Vérifier l'implantation des algorithmes d'arbitrage;
- Vérifier le fonctionnement de l'ensemble du système d'interconnexion.

La méthode de tests choisie doit permettre la réutilisation de certains bancs d'essais au fil de la conception. Le système d'interconnexion peut participer aux tests des différents modules qui y sont reliés. Aussi, le bus peut compléter les autres méthodes de tests, tel que le JTAG. Il est donc possible d'ajouter un troisième aspect au processus de vérification, à savoir l'ajout d'un mode « test » aux différents bus. Le fonctionnement du bus en mode test est décrit dans la spécification AMBA [3] et s'apparente à la conception en vue de la testabilité.

4.3.1 Tests applicables

La vérification du système est considérée avant tout comme un projet logiciel. L'ajout de composants matériels est cependant nécessaire afin d'améliorer la contrôlabilité et l'observabilité des tests lors de l'implantation aux niveaux d'abstraction inférieurs.

Différents bancs d'essais doivent être conçus pour réaliser la vérification fonctionnelle. Les bancs d'essais disposeront d'une visibilité allant de la « boîte noire » à la « boîte grise ». La vérification au niveau système a débuté lors de l'exécution du modèle UML et s'est poursuivie tout au long de l'intégration des modules. L'ajout de composants TVM simples (*Transaction Verification Modules*) lors des simulations de VHDL devrait optimiser l'application des tests en travaillant au niveau « transaction ».

La vérification structurelle quant à elle pourra être réalisée à l'aide d'outils de tests. La couverture de branches dans le code VHDL sera la principale méthode utilisée. Au niveau de la conception en vue du test, l'insertion d'une chaîne de balayage dans les modules appropriés sera effectuée lors des étapes de synthèse.

Chaque module a fait l'objet d'une vérification unitaire lors de sa conception. Cette vérification se limite habituellement à l'analyse des chronogrammes issus d'un banc d'essais simple.

4.3.2 Cas de tests

Cette section décrit les cas de tests du système. Les spécifications qui se rattachent à chaque cas de tests sont indiquées en annexe 1. Le test des interconnexions fut réalisés en quatre phases.

Tout d'abord, les bus AHB des processeurs ARM ont été testés de manière à s'assurer qu'ils respectaient la norme de communication AMBA. À ce stade, il est important de vérifier l'exécution du code de tests implémenté en C++, traduit en assembleur par le compilateur du ARM, qui permet la communication entre les modules et le processeur ARM à travers les bus AMBA.

Le bus de contrôle AHB 40 MHz a été testé avec les deux autres bus AHB des processeurs ARM. Puis, à partir d'un processeur ARM, nous avons exécuté des opérations de lecture et d'écriture vers les différents modules du système à travers le bus de contrôle AHB 40 MHz.

L'avant dernière étape a permis de vérifier la communication à travers le bus 320MHz. Le test consistait à écrire simultanément et lire, à l'aide des huit modules maîtres de ce bus, des données dans la mémoire principale. Ces opérations ont permis de valider le transfert des données et de vérifier s'il y avait pas contention dans ce bus.
En terminant, nous avons vérifié la connexion complète du système entre les processeurs ARM et la mémoire principale. Cette vérification permettait de valider la communication entre les deux domaines de bus à savoir le bus de contrôle AHB 40 MHz et le bus de données AHB 320 MHz.

L'ensemble de ces phases de tests vérifiaient les communications et les interconnexions entre les modules selon la norme AMBA.

4.3.3 Bancs d'essais

Le temps alloué ne permet pas l'utilisation de bancs d'essais évolués. Les bancs d'essais seront ainsi conçus en langage C et VHDL. Les bancs d'essais utilisés dans la vérification du système sont présentés en annexe 1.

4.3.4 Résultats

Les différents tests effectués ont été totalement satisfaisants. Dans le chapitre précédent, nous avons présenté une partie de ces tests dans la section co-vérification. Néanmoins, il serait intéressant de présenter les autres résultats pertinents obtenus lors de cette vérification.

4 .3.4.1 Test du bus AHB 320MHz.

La figure 4-19 présente le résultat automatique du test. Deux valeurs sont écrites et ensuite lues. La figure montre bien que les communications entre les modules et le bus AMBA fonctionnent adéquatement.

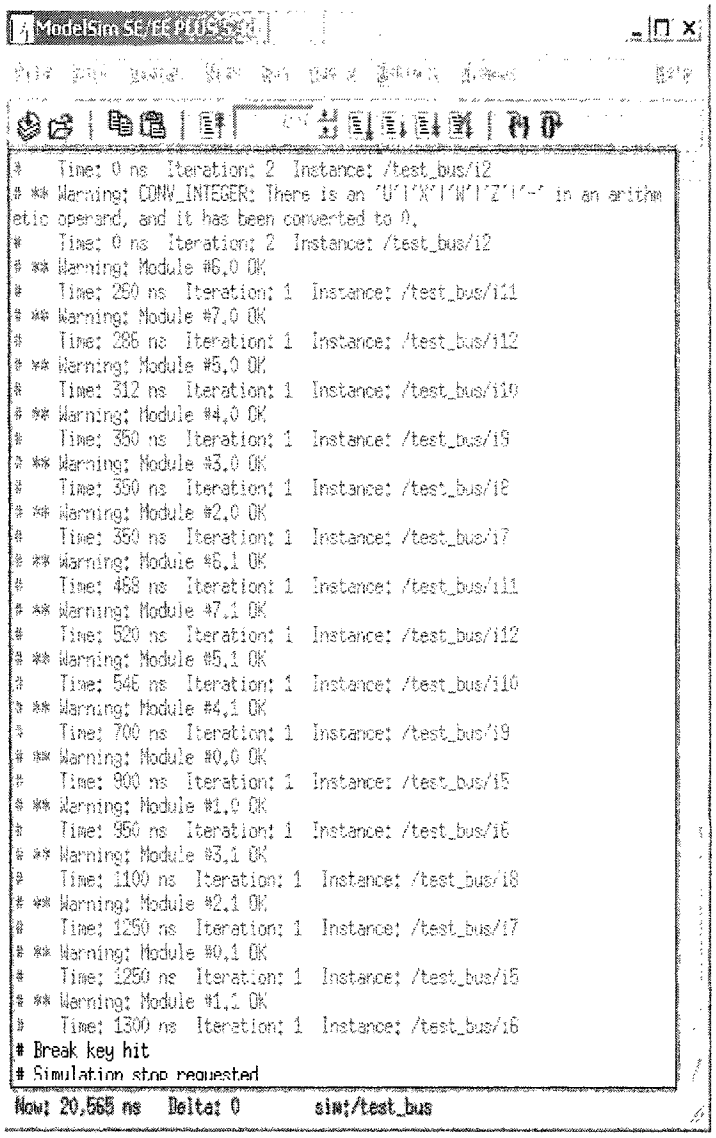

Figure 4-19 Test automatique du bus 320 MHz.

La figure 4-20 présente le bus AHB 320 MHz dont les phases d'adresses et de données créent un pipeline à deux niveaux. La première opération est effectuée par l'arbitre qui sélectionne le module qui a accès au bus. Le module sélectionné passe ensuite dans ses phases d'adresse et de donnée. La figure met en évidence le passage du jeton entre les huit modules.

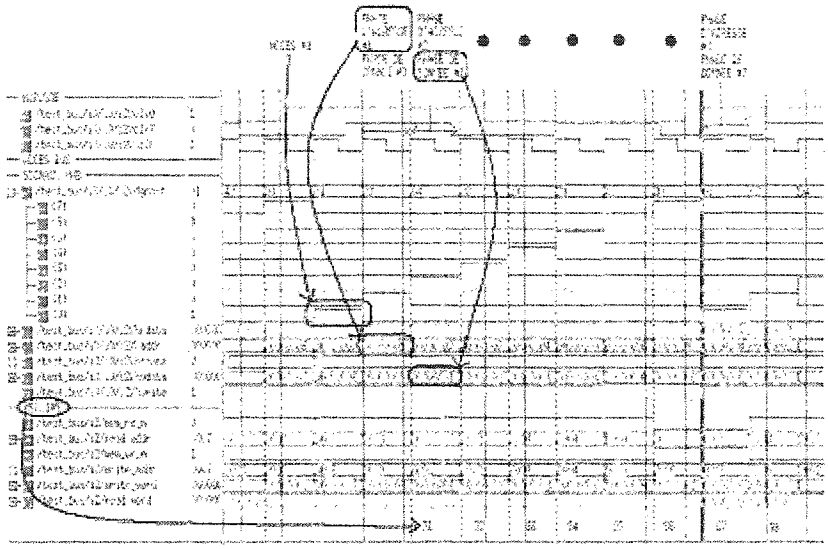

Figure 4-20 Illustration du bus AHB 320 MHz qui opère en pipeline.

La figure 4-21 présente les signaux côté mémoire. On constate que dans le cas d'une écriture, il n'y a aucun temps mort entre l'accès consécutif par deux modules.

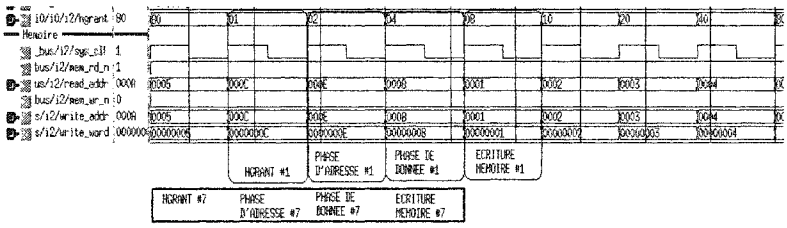

Figure 4-21 Mémoire en mode écriture.

La figure 4-22 présente également les signaux de la mémoire en lecture. On constate également qu'il n'y a pas de latence.

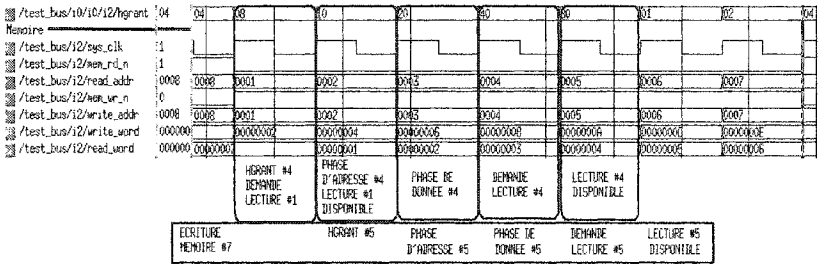

Figure 4-22 Mémoire en mode lecture

La figure 4-23 présente les requêtes et les réponses d'un module maître 40 MHz.

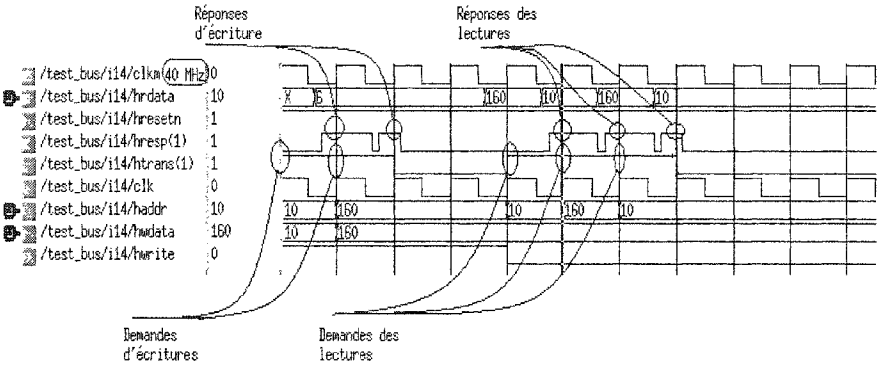

Figure 4-23 Module maître 40 MHz du bus 320 MHz.

Le module écrit deux valeurs consécutives, soit 10 et 160, suivi de deux lectures permettant la récupération des données. Il n'y a aucune latence avec un module de 40MHz. La lecture et l'écriture peuvent se faire en rafale. Le bus 320MHz utilise une simple communication pour effectuer chaque transfert.

La figure 4-24 présente les requêtes et les réponses d'un module maître 80 MHz.

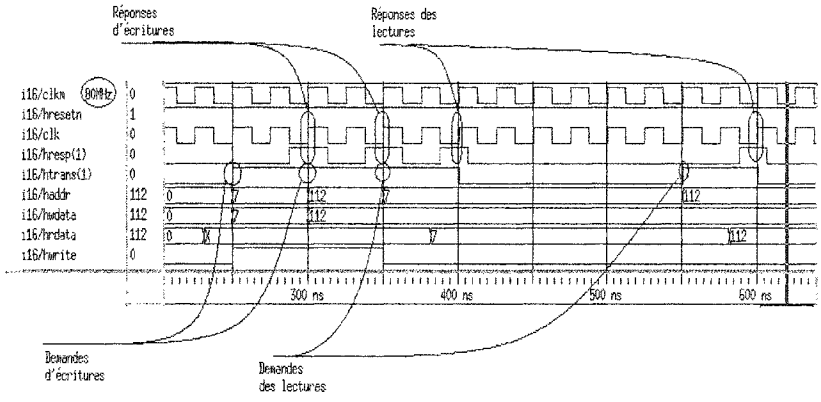

Figure 4-24 Module maître 80 MHz du bus 320 MHz.

Le module écrit deux valeurs consécutives soit 7 et 112. Ensuite, deux lectures permettant la récupération des données. Il y a une latence d'un cycle dans le module à

80MHz. Le transfert de données ne peut pas être effectué en rafale. Le bus 320MHz alloue seulement une tranche de temps par huit cycles d'horloge à chaque port .

*4 .3.4.2 **Synthèse du bus AHB 320MHz.***

Les contraintes de cette implémentation ont été extraites à partir de la technologie 0.18 micron CMOS de la bibliothèque de cellules normalisées et fournies par la CMC [6]. Afin de compléter le modèle comportemental du système, les délais et la surface de chaque bloc ont été obtenus à partir des outils de synthèse de Synopsys. La figure suivante montre la structure schématique du bus AHB 320 MHZ après synthèse.

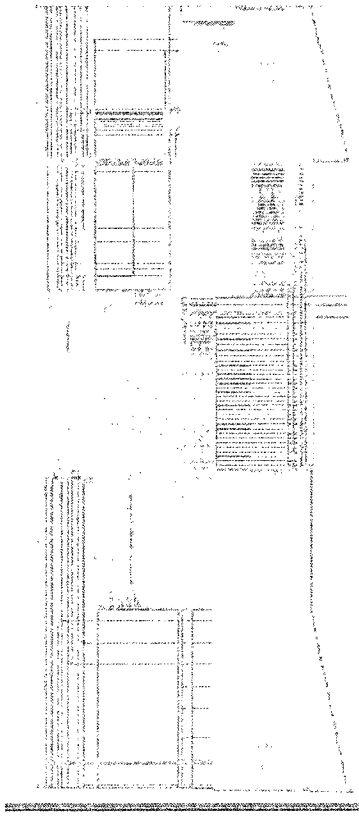

Figure 4-25 Synthèse du bus 320MHz

L'analyse de cette synthèse a permis de constater que l'objectif de performance de 320 MHz fixé pour un tel bus est réalisable. La fréquence de fonctionnement

approximative de ce dernier estimée par les outils Synopsys est de 657.89MHz avant placement et routage.

4 .3.4.3 Placement routage du bus AHB 320MHz.

La fonctionnalité du système après placement et routage a été vérifiée dans les simulations des bancs d'essais décrits précédemment. Les résultats ont été comparés à ceux obtenus par des simulations avant synthèse. Le placement et routage du bus a été fait à l'aide des outils de Cadence [7]. Nous pouvons voir dans la figure suivante le dessin des masques du bus à plus de 510.12 MHz après placement et routage.

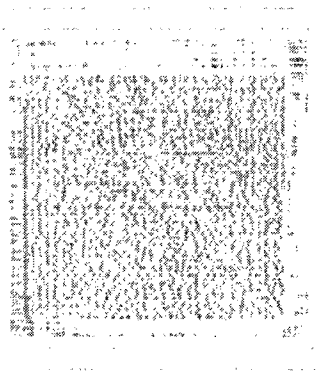

Figure 4-26 Dessin des masques du bus 320MHz

En conclusion le système d'interconnexion de la plate-forme SoC du convertisseur de protocoles a été implanté. L'implantation matérielle du bus AHB 320 MHz exige une surface de 481.8µm X 481.6µm qui correspond à 3413 portes avec la technologie CMOS 0.18 micron.. Cette cellule est capable d'atteindre une fréquence de 510.12 MHz.

CHAPITRE 5

CONCLUSION

5.1 SYNTHÈSE DES TRAVAUX

Dans ce mémoire, nous avons étudié quelques aspects de la convertibilité des protocoles de communication d'une manière générale. Après un survol de deux protocoles Firewire et Gigabit Ethernet, nous avons identifié un ensemble de propriétés qui les caractérisent et montré comment certaines peuvent entraver leur convertibilité. Notre application repose sur la conversion des protocoles Firewire et Gigabit Ethernet. Firewire est une nouvelle spécification capable de supporter de hauts débits comparables à ceux du « Gigabit Ethernet ». Nous avons souligné les principales différences existantes entre ces deux protocoles.

Une vue d'ensemble sur les différents systèmes existants permettant la conversion de certains protocoles a été présentée. Ceci nous a permis d'introduire et de classifier l'architecture courante de notre convertisseur de protocoles réseaux. Nous avons ensuite étudié certains mécanismes d'interconnexion utilisés dans les plates-formes SoC.

Les analyses faites lors de l'implémentation de l'architecture courante du convertisseur de protocoles sur la plate-forme d'évaluation Integrator A/P nous ont permis de constater qu'on pouvait alors gagner en performance et que notre modèle était moins flexible contrairement aux exigences d'une plate-forme SoC générique.

En effet, notre convertisseur engendre des délais supplémentaires de traitement des paquets. Ceci peut nuire dans un contexte de réseaux à hauts débits où certains flots de paquets transportent des données en temps réel. L'augmentation du délai de conversion peut aussi engendrer la perte des paquets lorsque les tampons de réception sont pleins. Pour y remédier, nous avons proposé une méthode d'analyse de conception de haut niveau.

Suite à une amélioration des spécifications du convertisseur, nous avons élaboré un modèle comportemental à haut niveau d'abstraction. Nous avons testé l'algorithme du modèle comportemental sur un ensemble de flots de paquets Gigabit Ethernet et Firewire. Les résultats obtenus ont confirmé l'efficacité de cette modélisation. De même, nous avons élaboré un modèle architectural en plusieurs étapes. Ceci nous a permis de proposer une nouvelle architecture de convertisseur de protocoles.

Le principe de cette nouvelle architecture repose sur une méthode d'interconnexion à bus partagés. Le protocole de communication utilisé par ces bus est basé sur la norme AMBA. La méthode suivie lors de la conception des bus AMBA a permis de développer le système par raffinements successifs. La première étape a permis l'identification des fonctions du système et des sous-blocs qui le constituent. Ensuite, chaque sous-bloc a fait l'objet d'une analyse détaillée. Un plan de vérification adapté à un environnement de bus a été mis sur pied. La vérification débute dès le modèle comportemental et s'est poursuivie lors des simulations après synthèses. Des règles de conception en vue du test telle que l'insertion d'un mode « test » dans les bus et l'application de chaînes de

balayage faciliteront la vérification de la plate-forme SoC générique du convertisseur de protocoles une fois tous les modules intégrés.

5.2 LIMITATIONS ET RECHERCHES FUTURES

Le modèle de convertisseur proposé doit être considéré comme un schéma de base qui peut être utilisé pour implémenter plusieurs convertisseurs de protocoles, et non un convertisseur complet et totalement générique disposant de toutes les fonctionnalités pour convertir des protocoles quelconques. Compte tenu de la complexité de la tâche, il n'est pas possible d'implémenter dans le cadre d'une maîtrise toutes les fonctions nécessaires à la conversion de tous les services couramment utilisés des protocoles Gigabit Ethernet et Firewire. Il faut aussi préciser que certains modules de notre implémentation ont été émulés dans ces tests, car ils étaient encore en phase d'élaboration conceptuelle.

Les tâches à accomplir à partir de maintenant sont la vérification à l'aide des bancs d'essais, la correction des erreurs de conception et l'analyse de performance du système une fois que tous les modules seront intégrés. Les résultats de cette vérification permettront de valider la fonctionnalité totale du système.

Des travaux futurs pourront être orientés sur la suite de l'étude des protocoles de communication réseau. Les algorithmes de traitements des données de vidéo numériques font partis du nouveau champ d'action envisagé. Ainsi, des études sur les algorithmes de transmission des données vidéo numériques existants seraient un élément de départ.

Sur le plan architectural, le niveau de flexibilité de la nouvelle plate-forme SoC est un avantage important pour la réutilisation d'une bonne partie ou de la totalité de ces modules. Ceci facilitera le temps de développement d'un nouveau système pour des applications de traitement de données vidéos numériques. La plate-forme SoC deviendra plus générique au fur et à mesure des itérations qui enrichiront cette structure de départ.

RÉFÉRENCES

[1] ALDWORTH, P. J.; "System-on-chip Bus Architecture for embedded Applications"; International Conference on Computer Design(ICCD'99),1999, pp. 297-298.

[2] AMD; Alchemy solutions AU1000; "A high-performance/low-power MIPS (SoC) 400 MHz at 0.5 watt''; http://www.amd.com/us-en/assets/content_type/DownloadableAssets/26328D_Alchemy_Au1000.pdf; Janvier 2003.

[3] ARM, Technical specification; "AMBA Specification''; Doc N0: ARMIHI-0011A, Issued : May 2001.

[4] ARM; "Integrator ASIC Platform development Boards (AP)''; ARM, 2001.

[5] CADENCE ; " VCC 2.1 Production documentation" ; 2001.

[6] CANADIAN MICROELECTRONIC CORPORATION; "Block Authoring Flow: Soft IP Block"; ICI-104 Version 1.1 November 7, 2002.

[7] CANADIAN MICROELECTRONIC CORPORATION; ''Tutorial on CMC's Digital IC Design Flow''; Document ICI-096, Part of tutorial Release V1.4, October 2002.

[8] CHANG, H.; COKE, L.; HUNT, M.; MARTIN, G.; McNELLY, A.; TOLDD, L.; "Surviving the SOC revolution''; Kluwer Academic Plublishers, 1999.

[9] COHEN, B.; "VHDL Coding Styles and Methodologies"; 2nd Edition, Kluwer Academic Plublishers, 1999.

[10] COHEN, B.; "Component Design by Example ... a Step-by-Step Process Using VHDL with UART as Vehicle"; ISBN 0-9705394-0-1, 2001.

[11] GHATTAS, H.; SAVARIA, Y.; ''Design dedicated low complexity embedded processors for SoC network processing applications''; NewCas, 2003.

[12] GUERRIER, Pierre ;, '' Un Réseau d'interconnexion pour systèmes intégrés '' , Thèse de l'Université Pierre et Marie Curie, Mai 2000.

[13] HALL, P.; "On Representation of Subsets"; Journal of the London Mathematical Society, vol. 10, pp. 26-30,1934.

[14] HENNESSY, J.L.; PATTERSON, D.A.; "Architecture des ordinateurs, une approche quantitative 2^{e} édition''; Thomson Publishing et Morgan Kaufman Publishers. France, 1996.

[15] IBM; CoreConnect Bus Architecture; http://www.chips.ibm.com/products/coreconnect; Consultée Aout 2002.

[16] IBM, ''The Network Processor'' ; http://www-3.ibm.com/chips/techlib/techlib.nsf/techdocs/852569B20050FF778525694A005B5B00/$file/networkwp.pdf , consulté en janvier 2003.

[17] IEEE, ''Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications'', IEEE Std 802.3, 1998.

[18] IEEE, ''IEEE Standard for high performance serial bus''; IEEE Std 1394, December 1995.

[19] INTEL; IXP1200 Network Processor Datasheet; ftp://download.intel.com/design/network/datashts/27829810.pdf; Décembre 2001.

[20] JOHNSON, E.; KUNZE A.; IXP1200 Programming (Sample Chapter); Intel Press; http://www.intel.com/intelpress/ixp1200/ixp-chapter.pdf; Février 2002.

[21] KAPOOR; ROHIT; "Network Processors"; http://www.cs.ucla.edu/~rohitk/NetworkProcessor.htm; Consulté en juillet 2002.

[22] KISHI, M.; TAKATSUKA, K.; NAKAZAWA, T.; "Hardware Development of μPLAT, Special Issue on SPA "; OKI Technical Review, OTR 184, December 2000, www.oki.com/en/otr ; consulté en janvier 2003.

[23] LEIJTEN, J.; MEERBERGEN, J. V.; TIMMER, A.; JESS, J.; "Stream communication between Real-Time tasks in a High-Performance Multiprocessor", Proceedings of the Design Automation and Test in Europe Conference, pp. 125-131,1998.

[24] MBAYE, M.; TOHIO, B.; SAVARIA, Y.; PIERRE, S.; ''Performance of Firewire-Ethernet Protocols Conversion on an ARM7 Embedded Processor''; CCGEI 2003, IEEE Canada.

[25] MENTOR GRAPHIC; "Seamless User's and Reference Manual. Software Version 4.0''; Seamless Hardware/Software co-verification, 2000.

[26] PEPGA BISSOU, J.; BA A.; BERTOLA, M.; DONZEL, G; NGONGANG, J.P.; PLANTE, P.; ''Conception d'un modèle exécutable d'un convertisseur de protocoles''; Rapport finale du cours VLSI 2, Hiver 2001.

[27] PEPGA BISSOU, J.; THOMAS, C.; SAVARIA, Y.; ''Integration of a Network Protocol Converter Using the ARM Integrator Platform"; Texpo 2002, CMC.

[28] PRESMAN, R.S.; "Software Engineering, A Parttitionner's Approach 5th Edition''; McGraw-Hill Higher Education, 2001.

[29] RYU, K. K.; SHIN, E; MOONEY, V. J.; "A comparison of five different multiprocessor SoC Bus Architectures''; Euromicro Symposium on Digital Systems Design (DSD'01), P.0202, September 2001.

[30] SALEH, R.; "System-on-Chip Trends Conference''; System-on-Chip Workshop 2001 Proceedings. Ottawa Canada.

[31] SHAH; NIRAJ; "Understanding Network Processors"; *http://www-cad.eecs.berkeley.edu/~niraj/papers/UnderstandingNPs.pdf*; Septembre 2001.

[32] SHAH, N.; PLISHKER, W.; KEUTZER, K.; ''NP-Click : Programming Model for the Intel IXP1200''; 2002.

[33] SANGIOVANNI-VINCENTILLI, A.; MARTIN, G.; "Platform-based design and software design methodology for embedded systems''; IEEE design and test of computers , 2001, pp. 23-33.

[34] SAVARIA,Y.; "Conception et vérification des circuits VLSI''; Éditions de l'École Polytechnique de Montréal, 1988.

[35] STAUNSTRUP, J.; WOLF,. W.; "Hardware/Software Co-Design, Principles and Practice''; Kluwer Academic Publishers, 1997.

[36] SYNOPSYS; ''DesignWare AMBA-based Microprocessor Sub-system, Speeding SoC Design with the Next Generation of Reuse"; Synopsys DW Star IP for MIPS seminar, 2001.

[37] TILMAN, Wolf; "Design of an Instruction Set for Modular Network Processors"; IBM T.J. Watson Research Center, Octobre 2000.

[38] TOHIO, B.; ''Aspects théoriques de la convertibilité des protocoles de communications''; Mémoire de maîtrise École polytechnique de Montréal, Mai 2003.

[39] VIRAGE; " Custom-Touch ASAP''; Virage Logic Corporation, 2000.

[40] WASHINGTON UNIVERSITY, Internet HDTV; http://www.washington.edu/hdtv/sc99/reference.html, Avril 2000.

[41] XU, J.; WOLF, W.; "Platform-based design and the first generation Dilemma"; Electronic Design Process (EDP), 2002.

ANNEXE

ANNEXE 1: CO-VÉRIFICATION DE LA PLATE-FORME SoC DU CONVERTISEUR DE PROTOCOLES RÉSEAUX

A1- 1 ENVIRONNEMENT DE CO-VÉRIFICATION

L'environnement de co-vérification est l'outil seamless. La figure suivante présente l'ensemble de la nouvelle architecture du convertiseur de protocoles réseaux. Dans cette figure, le module Screem (écran) a été ajouté pour l'affichage des résultat de tests afin de faciliter la vérification du fonctionnement du système.

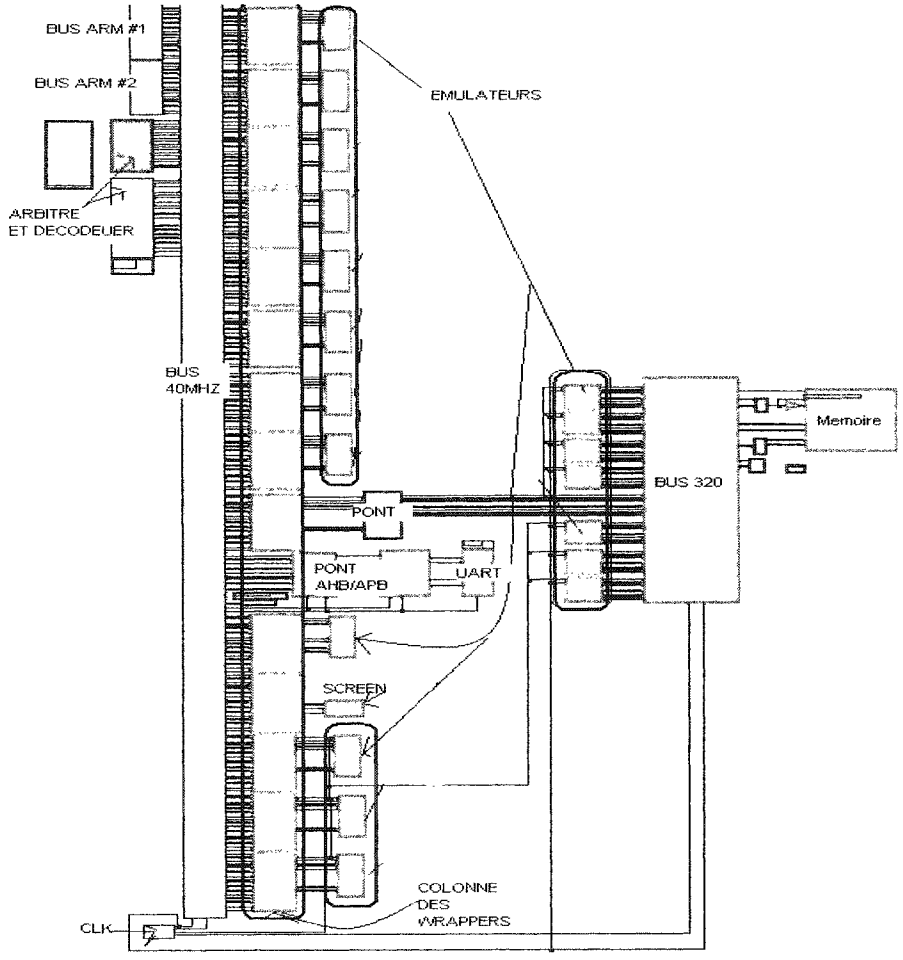

Figure A1-1 Structure de co-vérification de la plate-forme SoC

Les différents modules de cette structure sont dévéloppés en VHDL et la vérification du fonctionnement du système est faite grâce à l'application de test, dévoloppé en C++, exécutable via l'un des processeurs ARM..

A1-2 FONCTIONNEMET DES DIFFÉRENTS MODULES

Nous avons effectué des tests d'une demande d'écritures et de lectures par l'ensemble des modules du système. La figure A1-2 représente une demande d'écriture par un module 40 MHz au contrôleur d'interface. La lettre H est inscrite dans un module esclave. La phase d'adresse précise le module à sélectionner à partir de l'adresse (0x87000000) et précise s'il s'agit d'une lecture ou d'une écriture. La donnée est positionnée au prochain front d'horloge. On note que le module client est activé lorsque la donnée est présentée sur le bus. Ceci démontre l'utilisation d'une interface qui est nécessaire pour le décalage de l'adresse et de la donnée. Nous constatons que le système respecte la norme du protocole AMBA AHB.

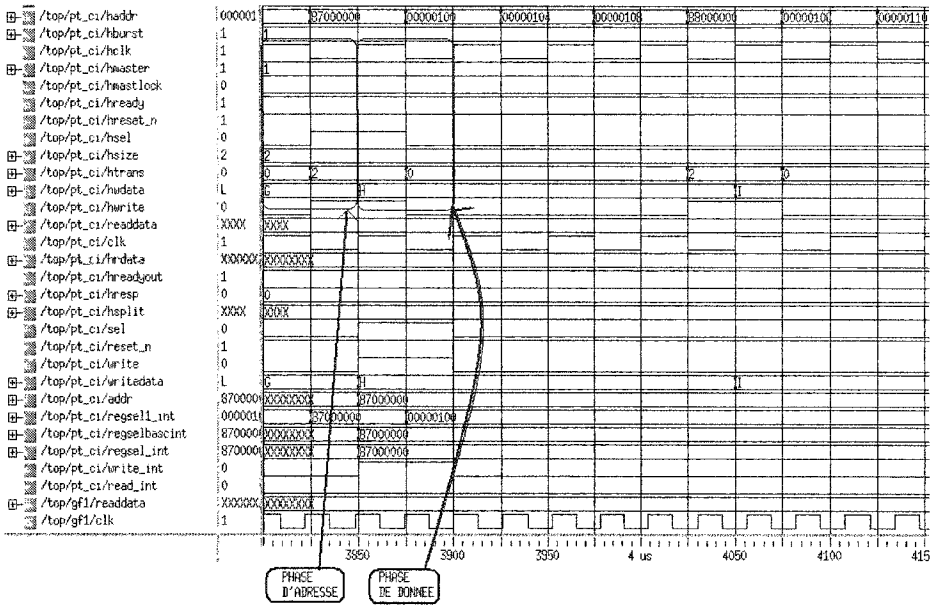

Figure A1-2 Exemple d'un module 40MHz en opération d'écriture

La figure A1-3 illustre la lecture de la lettre H écrite auparavant. Les phases d'adresses et de données sont identiques en écriture conformément à la norme AMBA.

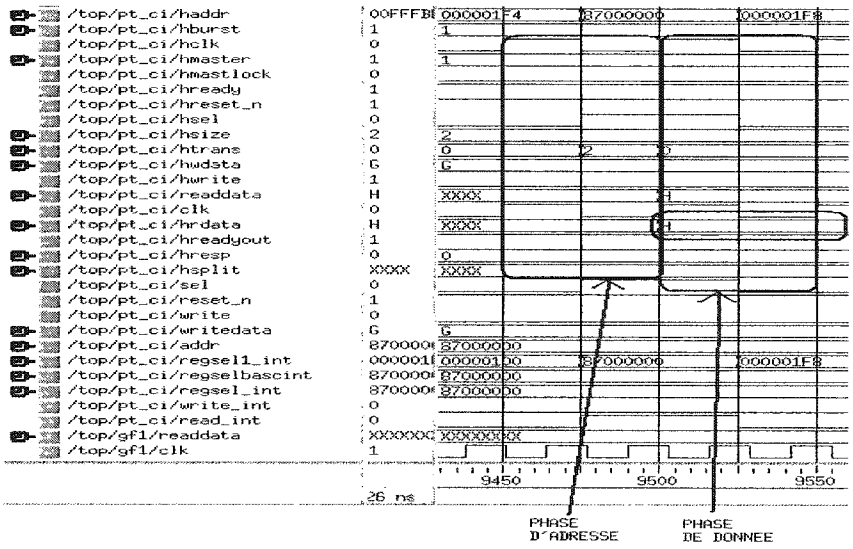

Figure A1-3 Exemple d'un module 40MHz en opération de lecture

Dans le cas des autres modules 80 MHz, le fonctionnement est le même que ceux des modules 40 MHz. La figure A1-4 présente un exemple d'écriture par un module 80

MHz. Il s'agit d'écrire la lettre L à l'adresse 8C000000. On remarque bien les deux phases du transfert.

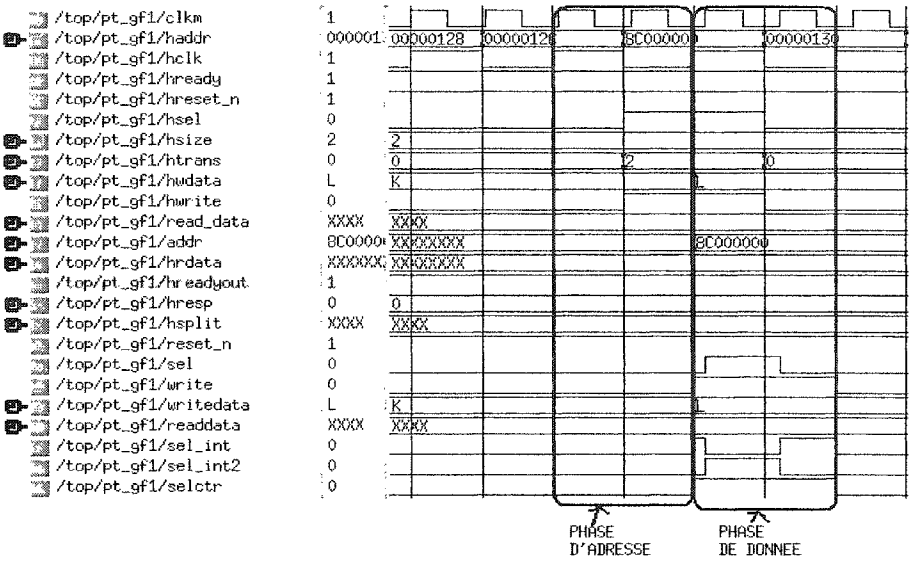

Figure A1-4 Exemple d'un module 80MHz en opération d'écriture

La figure ci-dessous illustre la lecture de la lettre L écrite auparavant à l'adresse 0x4C par un module 80 MHz. Les phases sont identiques à l'écriture.

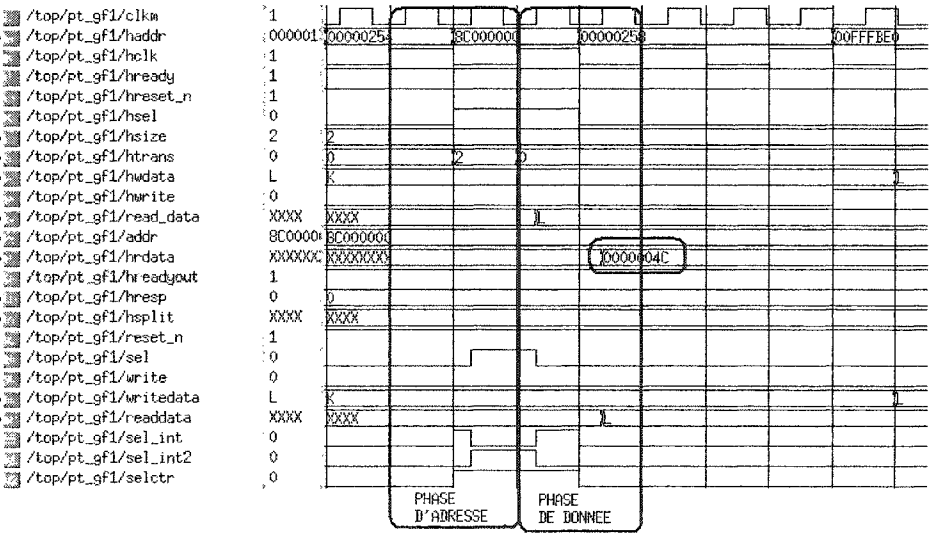

Figure A1-5 Exemple d'un module 80MHz en opération de lecture

La figure A1-6 représente le bon fonctionnement du pont AHB/AHB entre les deux domaines de bus. On remarque que le transfert des signaux de contrôle et des données d'un bus à un autre est fait en respectant la synchronisation requise dans chacun des domaines de bus.

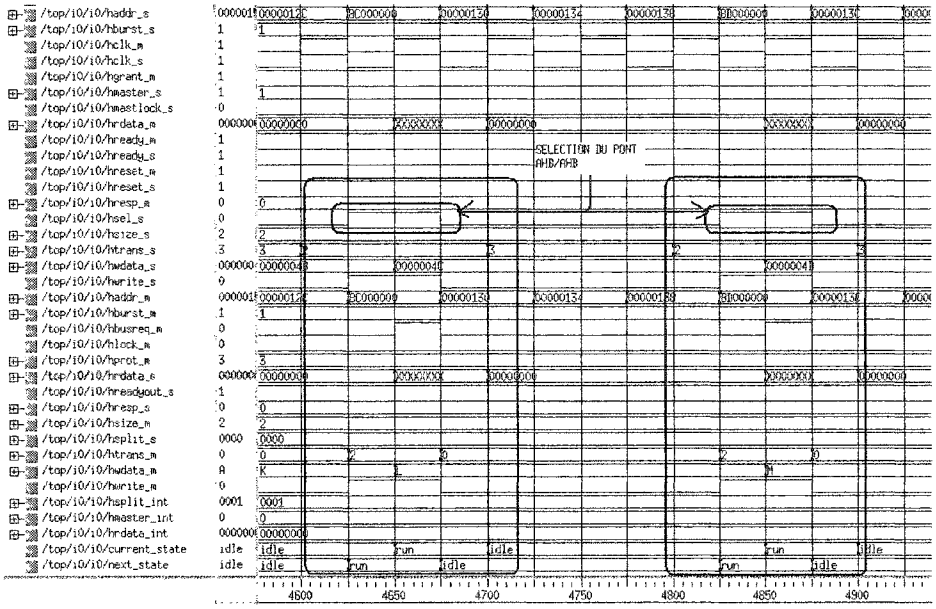

Figure A1-6 Exemple de synchronisation des deux domaines de bus AHB.

A1-3 CAS DE TESTS

Accès locaux conformes AMBA-AHB

Effectuer des requêtes lecture-écriture pour chacun des modules des deux bus.

Bancs d'essais behav_tb et trans_tb. TVM utilisés dans trans_tb : AHB Master, AHB Slave, AHB Monitor.

Accès inter-bus conformes AMBA-AHB

Accès lecture / écriture du bus de contrôle vers le bus de données.

Bancs d'essais behav_tb et trans_tb. TVM utilisés dans trans_tb: AHB Master, AHB Slave, AHB Monitor.

Accès conformes AMBA-APB via le pont AHB-APB

Requêtes lecture / écriture du bus d'un maître sur le bus de contrôle au UART

Accès lecture immédiatement suivi d'un accès lecture par le même maître

Accès lecture immédiatement suivi d'un accès lecture par deux maîtres

Accès écriture immédiatement suivi d'un accès lecture par le même maître

Accès écriture immédiatement suivi d'un accès lecture par deux maîtres

Bancs d'essais behav_tb et trans_tb. TVM utilisés dans trans_tb : APB Slave, APB Monitor.

Déroulement de l'arbitrage sur le bus de contrôle

Émettre des requêtes d'accès au bus à partir des deux maîtres AHB. Vérifier que l'arbitre alloue l'accès successivement aux deux maîtres.

Vérification fonctionnelle avec behav_tb. Simulation à l'aide d'émulateur maître / esclave VHDL et du TVM AHB Monitor dans le banc d'essais trans_tb.

Déroulement de l'arbitrage sur le bus de mémoire

Initialiser l'allocation round-robin du bus et vérifier si chaque maître se voit donner l'accès au bus. Banc d'essais perf_tb.

Mise en mode test

Mettre les bus AHB en mode test à partir d'un banc d'essais et du signal TBUS. Vérifier les états du *Test Controller Interface*. Vérifier l'écriture et la lecture de vecteurs de tests vers les modules.

Synchronisation des modules

Vérifier que les modules qui opèrent à des fréquences de 40MHz et 80MHz communiquent de façon transparente avec la mémoire 320MHz. Ce cas de test peut être réalisé par le banc de test unitaire des *adaptateurs maître et esclave*.

Cas limites sur les bus AHB

Il faudra revenir sur la définition des cas limites lorsque l'implantation sera plus avancée. Banc d'essais : trans_tb. \

A1- 4 BANCS D'ÉSSAIS

Le temps alloué ne permet pas l'utilisation de bancs d'essais évolués. Les bancs d'essais seront ainsi conçus en langage C et VHDL. Les bancs d'essais utilisés dans la vérification du système sont les suivants :

Banc d'essais du modèle comportemental (behav_tb)

Ce banc d'essais a permis la validation des spécifications face aux requis. Il est également utilisé dans la vérification fonctionnelle des cas de tests 1 à 5 décrits plus haut. Le banc d'essais est entièrement conçu en langage C++ et il génère et vérifie lui-même les vecteurs de tests. Les adresses et les données sont générées de façon pseudo-

aléatoire tandis que les types de transaction sont générés de façon déterministe. Des éléments de ce banc d'essais peuvent être réutilisés dans le banc d'essais 2 ci-dessous.

Banc d'essais des transactions (trans_tb)

Ce banc d'essais doit être versatile pour permettre la vérification des cas de tests 1 à 4 de même que des cas 6 et 8. Il s'agit essentiellement de contrôler les trois TVM (Master, Slave et Monitor) afin de produire les conditions du cas de tests. Une analyse des résultats recueillis par le AHB Monitor permettra la vérification visuelle des réponses. Si le temps le permet, une vérification en temps réel avec le modèle de référence sera mise au point.

Banc d'essais de l'arbitrage haute performance (perf_tb)

Ce banc d'essais est dédié à la vérification de l'arbitrage sur le bus de la mémoire (cas de test 5). Il est constitué d'un ensemble d'émulateurs et d'un TVM AHB Monitor. Le banc d'essais devra permettre le fonctionnement du système pour une durée indéterminée afin de s'assurer de la stabilité de l'algorithme d'arbitration. Des métriques de performance devront également être extraites de ce banc d'essais afin de valider les fréquences d'opérations.